In: Feelders (ed.), Proceedings 10th BENELEARN meeting, p. 101-109, 2000

Comparing Bagging and Boosting for Natural Language Processing Tasks: A Typicality Approach.

Véronique Hoste Walter Daelemans *
CNTS - Language Technology Group, University of Antwerp
Universiteitsplein 1, 2610 Wilrijk
{hoste|daelem}@uia.ua.ac.be

Abstract

We report on a comparison of two well-known ensemble combination methods, bagging and boosting, in the context of a natural language processing task. In comparative studies on general Machine Learning benchmark tasks, boosting has been shown to be more sensitive to noisy data than bagging. Because of the special nature of language learning tasks (many subregularities and exceptions indistinguishable in general from noise), it is useful to carefully compare the strengths and weaknesses of both methods when applied to natural language problems. In the experiments, we focus on learning part of speech tagging with the decision tree algorithm C5.0. We analyze the error of bagging and boosting by correlating the typicality of the instances with the predictability of the combination methods. Results show that the resampling in boosting indeed results in a relatively lower error on exceptional cases as compared to bagging, but with a higher error on regular cases. We focus next on how to benefit from the typicality information of the training instances as an alternative way to produce the members of an ensemble, and show that this approach leads to generalization accuracies in the range of boosting.

1 Introduction

During the last few years, considerable effort has been invested by the Machine Learning community in improving the accuracy on classification tasks. Using an ensemble of classifiers rather than one single classifier has become a popular approach to achieving this goal. The underlying idea is that, when the errors made by the individual classifiers of an ensemble are uncorrelated to a sufficient degree and their error rate is low enough, the resulting combined classifier will perform better than any of the members of the ensemble. See e.g. Dietterich (1997) for an overview.

One possible ensemble method is to combine different types of learning paradigms. Classifiers trained by different types of learning algorithm can be combined by using voting or stacking (feeding the outputs of a number of classifiers as features for a second-level learner, also called cascading). This approach has gained some popularity in natural language processing (NLP) as well. Examples include Van Halteren et al. (1998) and Brill and Wu (2000) for part of speech tagging, Hoste et al. (2000) for grapheme to phoneme conversion, and Tjong Kim Sang et al. (2000) for the identification of base noun phrases.

Another possible approach is to use a single learning algorithm to obtain each individual classifier of the ensemble. In that case, the training examples may be manipulated to generate multiple hypotheses. The learning algorithm is run several times, each time with a different subset of the training examples. The composition of these subsamples depends on the approach being used. In the case of "bagging" (Breiman, 1996), each classifier is trained on a random selection of the training set. In "boosting" (Freund and Schapire, 1996), previously misclassified examples are assigned more weight than correctly classified examples in the next sample. In comparative studies on general Machine Learning benchmark tasks, boosting has been shown to generally outperform bagging, but to be more sensitive to noisy data (Dietterich, 2000; Bauer and Kohavi, 1999; Opitz and Maclin, 1999). In NLP research, boosting has been applied to improve output quality in tagging and PP-attachment (Abney, Schapire, and Singer, 1999), text filtering (Schapire, 1999), text categorization (Schapire and Singer, 2000), etc.

*This research was partially funded by the FWO project Linguaduct

In this paper, we will examine whether bagging and boosting always outperform a single classifier when applied to a natural language processing task, and compare the two approaches. The motivation to redo this comparison specifically for NLP tasks is that previous research has shown that most NLP tasks are highly disjunctive, i.e., instance space consists of many small regions and classes are polymorphous (the same class occurs in different regions in instance space). In other words, NLP task training sets contain many subregularities and exceptions not easily distinguishable from noise (Daelemans, Van Den Bosch, and Zavrel, 1999).

Our initial hypothesis is that the resampling in boosting will result in a relatively lower error on exceptional cases compared to bagging, but with a higher error on regular cases because of overgeneralization of the exceptions. As a test case, we applied the decision tree learner C5.0 (Quinlan, 1993) and the combination methods bagging and boosting to a Brazilian Portuguese tagging corpus. Part of speech tagging can be described as a process in which morphosyntactic categories (part of speech tags) are assigned to words based on lexical information about the tags co-occurring with the words, and contextual information about the syntactic environment in which the words occur in a text. Automatic part of speech tagging is useful in applications such as machine translation and information retrieval.

In a first group of experiments, we examined whether bagging and boosting improve the accuracy of the C5.0 decision tree learning algorithm. In order to have a more detailed view on the performance of both combination methods, we have computed for each instance its "typicality" (opposite of exceptionality). As expected, boosting performs better on exceptional, low-typical cases, whereas the normal C5.0 and bagging perform better on the more typical cases. As a consequence of these observations, we have investigated in a second set of experiments whether training different classifiers on groups of instances, representing different typicality ranges, can lead to more specialized, more accurate predictors.

In the following section, we explain our experimental setup, describing the data sets and classification methods being used in the different experiments. In Section 3, we investigate the predictive power of the classifiers in relation to the typicality values of the instances. In Section 4, we explore the use of typicality as a grouping technique. In Section 5, we summarize and conclude.

2 Experimental setup

2.1 Data set

The experiments reported here, were performed on part of a Brazilian Portuguese tagging corpus (Aires et al., 1999), which contains sentences extracted from textbooks and newspapers. From this corpus, which contains 51,775 instances, a data set was constructed. Since many words are ambiguous with respect to their part of speech tags, the local context of the words is used to select the most likely morphosyntactic category. All instances contain information about the focus word (the word to be tagged) itself (W), the part of speech tag of the word one and two positions to the left of the focus word (labeled d for disambiguated already), a label expressing the possible part of speech tags of the focus word (f), and of the words one and two positions to the left of the focus (labeled a for still ambiguous), and the third last (s), second last (s) and last letter (s) of the focus word. The part of speech tagging task can be defined as the conversion of fixed-size instances representing the focus word (W), with its context to a class representing the syntactic category of the focus word. The construction of the instances assumes a tagger that processes the input sentences from left to right (see Daelemans et al. (1996)). The words to the left of the focus word are already tagged and represented by their disambiguated part of speech tag, whereas the focus word itself and the words to its right are represented by a so-called ambiguity class, e.g. adjective-or-adverb.

Table 1: An example instance in the data set.

Attributes									class
W	d	d	f	a	a	s	s	s	
iniciarmos	AG	AG	AV	AJ	AL	m	o	s	AV

2.2 Method

In the base experiments, 10-fold cross-validation (Weiss and Kulikowski, 1991) was used as the experimental method for error estimation, with a 90% training set and a 10% test set. In the combination experiments, the 90% training set was again split into 10 equal parts and the combiner taggers were trained on 9 parts en tested on the remaining part. These results were concatenated to train the combined learners. For all experiments, C5.0 (Quinlan, 1993) was used as the baseline predictor. C5.0, which is a commercial version of the C4.5 program, generates a classifier in the form of a decision tree. This decision tree is used to classify a case by starting at the root of the tree and then moving through the tree until a leaf node (associated with a class) is encountered.

In the bagging (bootstrap aggregating) algorithm, proposed by Breiman (1996), a number of bootstrap replicates (with replacement) are built from the training set, producing a number of new training sets. In our experiments, 10 bootstrap samples were made from the data set. A majority vote on the results of the predictors trained on these learning sets then leads to the overall predictor.

For the boosting experiments, the adaptive boosting ("Adaboost") algorithm (Schapire, 1999) included in C5.0 (Quinlan, 1993), was used. As in bagging, several classifiers are generated starting from the training set. As a first step, a single classifier is trained and all weights associated with training instances are set equally. This classifier, however, makes mistakes and in the following step the weights of the incorrectly classified examples are increased so that the classifier is forced to focus on these examples. The errors committed by this second classifier receive more weight during the following round and this process of increasing the weight of incorrectly classified examples continues until a pre-determined number of iterations is reached. For our experiments, we have chosen 10 boosting rounds, which is the default number of trials used in C5.0.

The results in Table 2 show that both the boosted C5.0 and the bagged C5.0 outperform the single C5.0 classifier on our data set. A comparison of the results shows that boosting C5.0 leads to an error reduction of 12.3% compared to the base C5.0 classifier, whereas bagging performs 5.9% better. When we focus on the results of both combination methods on the data set, we see that boosting reduces the error of bagging with 6.7%.

Table 2: Accuracy of C5.0, bagged C5.0 and boosted C5.0 on the data set.

C5.0	Bagged C5.0	Boosted C5.0
94.94%	95.24%	**95.56%**

In order to improve our understanding of the strengths and weaknesses of both combination methods, we have analyzed the baseline C5.0 learner and its bagged and boosted versions in relation to their handling of more or less typical instances. This should give us a more detailed view on the positive and negative effects of bagging and boosting C5.0 for our natural language data set.

3 Typicality

The "typicality value" of a given instance denotes its grade of typicality or exceptionality. For the computation of an instance's typicality value, we have adopted the definition of Zhang (1992). In that paper, the family resemblance hypothesis of Rosch and Mervis (1975) is adapted. This hypothesis states that the most prototypical members of categories are those with most attributes in common with other members of that category and those with least attributes in common with other categories. Zhang measures the typicality of an instance as the ratio of its intra-concept similarity (average similarity to instances with the same class) to its inter-concept similarity (average similarity to instances with different classes). Instances with a larger intra-concept similarity can thus be described as typical instances, whereas instances with a larger inter-concept similarity are less typical. The instances of a data set can be divided in three different typicality groups:

- Instances with a typicality of less than 1 are more typical for other categories than for their own category and can thus be described as the exceptional cases.
- The instances with a typicality value close to 1 denote the boundary cases.
- Instances with a typicality value much higher than 1 are the typical cases.

In work by Daelemans et al. (1999), typicality is used as an editing technique for the data set in order to investigate whether it is beneficial for the generalization accuracy of a memory-based classifier to keep exceptional instances in memory. In this paper, we related the performance of the C5.0 base line predictor and the combination methods bagging and boosting with the typicality values of the instances in order to obtain a more detailed view on the strengths and weaknesses of the different systems. In the Brazilian Portuguese tagging corpus we used for the experiments, the typicality values vary from 0.12 to 50.82. In Figure 1, the number of correctly classified and misclassified instances is plotted against the typicality values present in the data set.

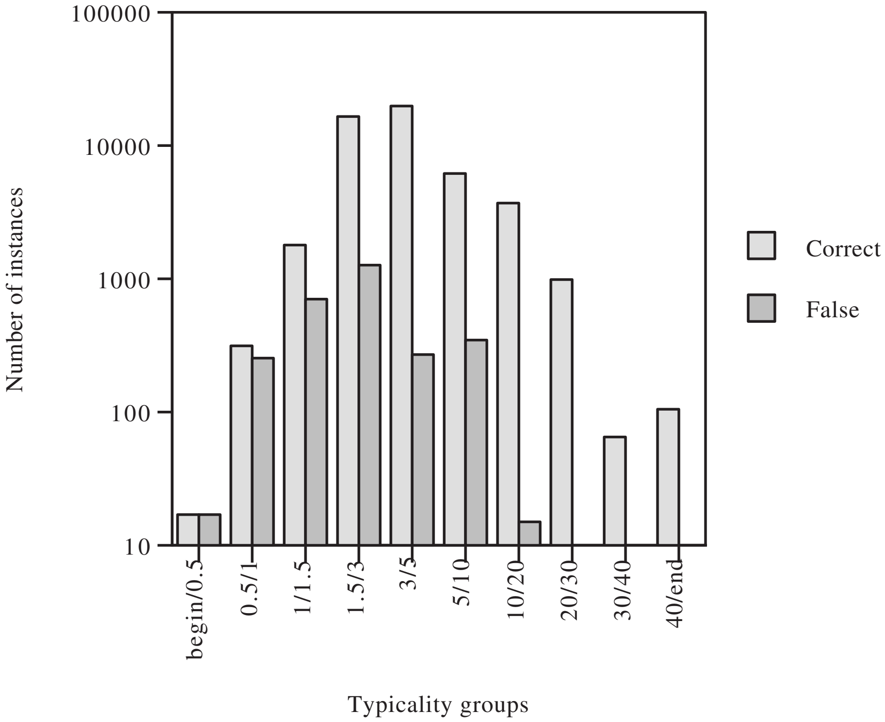

Figure 1: Number of correctly classified and misclassified instances for 10 different typicality ranges. The typicality values start in the range 0.0-0.5, denoting the exceptional cases, pass by the boundary cases with values close to 1 and end with the typical cases. For each typicality range, the number of correctly (left) and incorrectly (right) classified cases is given.

Figure 1 shows that especially the typicality values within the range of 1.5 and 5 are present in the data set. A closer look at the tagging accuracy of C5.0 for each of these ranges tells us that a little more than half of the instances in the typicality range 0.0-1 are correctly classified by C5.0. This is not surprising, since this range includes the more exceptional cases in the data set. Classifying the boundary cases (1-1.5) leads to a 71.79% tagging accuracy. We can state that there is a lot of space for improvement of tagging accuracy for the exceptional and boundary cases. Figure 1 also shows that the predictive power of C5.0 increases (100% accuracy in the range 20-end) when typicality increases.

In Table 3, the generalization accuracy of the C5.0 base line predictor, the bagged C5.0 and boosted C5.0, is displayed in relation with the typicality values of the instances. This is done by ordering the data set according to the typicality values of the instances. The predictions of the three classification methods are then divided in equal groups, in which the first instance of the first typicality group has the lowest typicality value (0.12), whereas the last instance of the last typicality group has the highest typicality value (50.82). The subdivision of the data set into different typicality groups (two, three and five) gives us a more subtle view on the predictive power of the three classifiers. Table 3 reveals that applying bagging to C5.0 does not make a convincing case. Although bagging (95.24%) is better than the base line C5.0 decision tree learning algorithm (94.94%), it is never the best prediction method in any of the typicality groups. When dividing the data set into two, three or five typicality groups, it becomes clear that the C5.O classifier and the boosted C5.0 have complementary strengths. As expected, boosting

Table 3: Accuracy of C5.0, bagged C5.0 and boosted C5.0 in relation to the typicality values of the instances.

Classification method	Generalization accuracy (%) on different typicality groups. The best performing classifier is indicated in bold.			
	1	2	3	5
C5.0	94.94	90.98 **98.90**	87.77 98.12 **98.93**	82.07 96.40 98.29 **98.88** **99.06**
Bagged C5.0	95.24	91.67 98.81	88.49 98.43 98.81	83.01 96.90 98.58 98.75 98.97
Boosted C5.0	**95.56**	**92.39** 98.73	**89.43** **98.59** 98.67	**84.18** **97.42** **98.78** 98.66 98.77

C5.0 performs best on the low-typical instances, since in boosting the learner is forced to focus on the hard examples by increasing the weights of the incorrectly classified instances in each boosting round. However, boosting C5.0 loses predictive power when classifying the group containing the high-typical, more regular instances. The best classifier for this group is the base line C5.0 classifier.

In Figure 2, we concentrate on the decrease of generalization accuracy of boosting C5.0 as opposed to the accuracies of C5.0 and its bagged version. In order to do so, we have divided the data set into three typicality groups and focused on the last two groups. The intersecting lines indicate the tendency of boosting to misclassify typical cases, whereas the C5.0 base line predictor performs better on these.

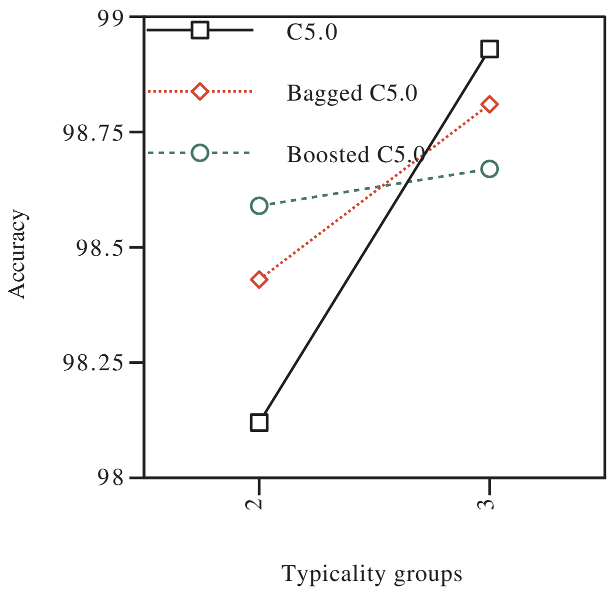

Figure 2: Tagging accuracy of C5.0, bagged C5.0 and boosted C5.0 in correlation with the typicality values of the instances (detailed view)

The differences in the generalization accuracies of the three classification methods in relation to the typicality values of the instances in our data set, indicate that taking into account the exceptionality-measure typicality to train different classifiers can be an interesting option. In order to benefit from the typicality information given for each training instance, we have investigated whether training classifiers on groups of instances, representing different typicality ranges, can lead to more accurate ensembles.

4 Typicality combination

In the previous section, we observed that typicality is a good measure to determine the exceptionality of instances. The correlation of the accuracy of C5.0 with the typicality values of the instances in Table 3 and Figure 2 shows that boosting C5.0 is more promising than the C5.0 base line predictor and its bagged version for the more exceptional and boundary cases. However, when predicting the more typical, more regular cases, the C5.0 base line predictor and the bagged C5.0 (which underperform for the first two typicality groups) outperform boosting. As a consequence of these observations, we have investigated in a new set of experiments whether it was possible to (i) keep or improve the accuracy level of boosted C5.0 for the group containing the low-typical instances and (ii) to keep or improve the accuracy of C5.0 for the group with the more regular, more typical instances by training classifiers on these groups. The approach we used in our experiments can be described in five steps: (i) calculate the typicality values of the instances in the training set, (ii) determine which is the best division of the training set into typicality groups, (iii) train n classifiers on n typicality groups, (iv) predict the typicality group of the test instances on the basis of the typicality information given in the train set, (v) apply voting or a stacked classifier on the base instance, the n classifications of the ensemble, and the predicted typicality. We will call this approach typicality combination.

In a first step, we have investigated which was the optimal number of typicality partitions of the train set, by training the C5.0 decision tree learner on two, three, five and ten groups, leading to different classifiers, all specialized in classifying instances within a certain typicality range. E.g., when dividing the train set into two typicality groups, we end up with a classifier which is more specialized in the harder cases and a second classifier which focuses on the more typical cases. In order to do so, the train set, which is sorted according to the typicality of the instances, was divided into equally large groups representing different typicality ranges (the first instance of the first group has the lowest typicality (0.12) and the last instance of the last group has the highest typicality (50.82)). The typicality values were not included as features in the training pattern. The C5.0 decision tree learner was trained on these training sets. This resulted in two, three, five or ten classifications, one for each classifier trained on each typicality group. By focusing on the exceptionality or typicality of instances, these classifiers can give more specialized predictions. In order to decide which classifier was the best predictor for a given instance group, we performed a special type of voting on the classifications of the specialized classifiers, in which the typicality value of the test item is taken into account. E.g., if that typicality value falls within the range of typicality group x, the prediction of the classifier trained on this typicality group x is retained. The results of the voting experiments for C5.0 are displayed in Table 4.

Table 4: Performance ceiling when doing voting over C5.0 classifiers trained on two, three, five or ten typicality partitions of the train set.

Number of typicality groups			
2	3	5	10
95.98	96.21	**96.38**	96.34
		90.48 **99.17** **99.50**	

The results in Table 4 clearly show the usefulness of using typicality information as a method to divide the train set into subsets. Training classifiers on these subsets thus leads to more specialized predictors. A comparison of these voting results with the results displayed in Table 3, reveals that all voting experiments outperform the C5.0 base line predictor, and also the bagged and boosted C5.0. A more detailed analysis (division into three typicality groups) of the voting results on the classifiers which are trained on five different typicality groups in relation to the results in Table 3, shows that the voting approach outperforms the boosted C5.0 for the low-typical cases and C5.0 for the high-typical cases.

However, since typicality is calculated on the basis of the relation between a given instance and its classification, the exact typicality value of a test instance is not known. Therefore, the typicality of a test case was predicted on the basis of the typicality values of the training instances. Instead of predicting the detailed typicality value for each test instance, we grouped the typicality values of the training instances into five typicality groups and for each test instance it was predicted whether its class was 1, 2, 3, 4 or

5, by using the memory-based learner TiMBL. TiMBL [1] (Daelemans et al., 2000) is a software package implementing several memory-based learning (lazy learning) techniques. *Memory-based learning* is a learning method which is based on storing all examples of a task in memory and then classifying new examples by similarity-based reasoning from this memory of examples.

Both the output of the classifiers trained on the 5 typicality groups and the predictions of the typicality group of the test instances lead to an enriched input pattern. In a first experiment, we performed a voting: the predicted typicality group (one to five) of each test item was taken into account and the prediction of the classifier trained on this typicality group was used. In a second group of experiments, two different learners, a decision tree learner and a memory-based learner take the enriched instances as input.

Table 5: Results of voting and combined C5.0 and MBL in relation to the C5.0 base line experiments.

	Classification method	Generalization accuracy on data set	Generalization accuracy on 3 different typicality groups		
			1	2	3
No typicality combination	Baseline C5.0	94.94	87.77	98.12	98.93
	Bagging C5.0	95.24	88.49	98.43	98.81
	Boosting C5.0	95.56	89.43	98.59	98.67
Typicality combination	Voting	95.16	88.94	98.27	98.27
	C5.0	95.22	88.68	98.44	98.53
	MBL	**95.62**	89.76	98.60	98.51

A comparison of the overall accuracy of the baseline C5.0 decision tree learner and the typicality combination results displayed in Table 5 reveals that the latter performs better on our task. Voting over the classifications of the five decision tree learners using estimated typicality of test instances leads to a 4.7% error reduction compared to the C5.0 baseline predictor. This result however, is nowhere near the ceiling performance of 96.38% (Table 4). The results of typicality combination with stacked learning reveal that all learners outperform C5.0 on our part of speech tagging task. Training C5.0 on the enriched input pattern, containing the output of the classifiers trained on the 5 typicality groups and the predictions of the typicality group of the test instances, leads to a generalization accuracy of 95.22%, which is similar to the accuracy of bagging C5.0 (95.24%). In the experiments where we use memory-based learning as a stacked learner, we can observe that it even outperforms the boosted C5.0 (95.56%). If we correlate the results of the voting and the ensemble experiments with the typicality values (three groups) of the instances, the following observations can be made: (i) the results of C5.0 and bagged C5.0 for the groups containing the low-typical and medium-typical instances can be improved. Moreover, the MBL experiment reaches a higher accuracy level than boosted C5.0 for these groups, and (ii) In the third typicality group, all results remain below the 98.93% tagging accuracy of C5.0. The C5.0 and MBL combined classifiers are the best performing classifiers in this group (around 98.5%).

5 Conclusion and Future Work

In this paper, we compared the performance of two well-known ensemble combination methods, bagging and boosting, in the context of a natural language processing task. A more detailed view on the performance of both combination methods and a baseline decision tree learner (C5.0) was obtained by linking the predictive power of the methods with the typicality values of the instances. These results reveal that boosting performs better on exceptional, low-typical cases, whereas the normal C5.0 and bagging perform better on the more typical cases.

In a second step of experiments, we investigated whether it was possible to benefit from the typicality values of the training instances by training different classifiers on groups of instances, representing different typicality ranges (typicality combination). The hypothesis was that this would lead to more accurate predictors. Results indicate that using typicality to group the training data can lead to a gain in classification accuracy. In our experiments, all methods using typicality outperform the base line C5.0,

[1] TiMBL can be downloaded from http://ilk.kub.nl

and the stacked learners are better than the bagged C5.0. Within the same set of experiments, the memory-based stacked learner achieves a better score than the boosted C5.0. However, our hypothesis that it was possible to (i) keep or improve the accuracy level of boosted C5.0 for the group containing the low-typical instances and (ii) to keep or improve the accuracy of C5.0 for the group with the more regular, more typical instances by training classifiers on these groups, is only partly confirmed. In the experiments indicating the ceiling performance of the C5.0 algorithm when performing voting, there was a strong indication that our hypothesis is viable. In the final experiments, this tendency can also be observed for the first two typicality groups (generalization accuracy close to that of boosted C5.0), but to a lesser extent in the third typicality group, where the base line C5.0 still outperforms all other methods.

We interpret the results presented in this paper to indicate that using typicality-bagging to train more accurate, more specialized classifiers can be an alternative to combination methods such as bagging and boosting. In future research, we will further explore the usefulness of this method on other natural language processing tasks. Furthermore, we will investigate in new experiments how typicality of test instances can be more accurately estimated.

6 References

Abney, S., R.E. Schapire, and Y. Singer. 1999. Boosting applied to tagging and pp-attachment. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Aires, R.V.X., S.M. Aluíso, D.C.S. Kuhn, M.L.B. Andreeta, and O.N. Oliveira Jr. 1999. Combining multiple classifiers to improve part of speech tagging: A case study for brazilian portuguese.

Bauer, E. and R. Kohavi. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–142.

Breiman, L. 1996. Bagging predictors. *Machine Learning*, 24:123–140.

Brill, E. and J. Wu. 2000. Classifier combination for improved lexical disambiguation. In *Proceedings of the Joint Seventeenth International Conference on Computational Linguistics and Thirty-sixth Annual Meeting of the Association for Computational Linguistics*, pages 191–195.

Daelemans, W., A. Van Den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–42.

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. Mbt: A memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop of Very Large Corpora*, pages 14–27.

Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 2000. Timbl: Tilburg memory based learner version 3.0 reference guide. Technical report-ilk 99-01, Induction of Linguistic Knowledge, Tilburg University.

Dietterich, T.G. 1997. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136.

Dietterich, T.G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):1–22.

Freund, Y. and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth National Conference*, pages 148–156.

Hoste, V., W. Daelemans, E. Tjong Kim Sang, and S. Gillis. 2000. Meta-learning for phonemic annotation of corpora. In *Proceedings of the Seventeenth International Conference on machine Learning*, pages 375–382.

Opitz, D. and R. Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Quinlan, J.R. 1993. *C4.5: programs for machine learning*. San Mateo: Morgan kaufmann Publishers.

Rosch, E. and C.B. Mervis. 1975. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605.

Schapire, R. E. 1999. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406.

Schapire, R.E. and Y. Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Tjong Kim Sang, E.F., W. Daelemans, H. Déjean, R. Koeling, Y. Krymolowski, V Punyakanok, and D. Roth. 2000. Applying system combination to base noun identification. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 857–863.

Van Halteren, H., J. Zavrel, and W. Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the Joint Seventeenth International Conference on Computational Linguistics and Thirty-sixth Annual Meeting of the Association for Computational Linguistics*, pages 491–497.

Weiss, S. and C. Kulikowski. 1991. *Computer systems that learn*. San Mateo, CA: Springer Verlag.

Zhang, J. 1992. Selecting typical instances in instance-based learning. In *Proceedings of the International Machine Learning Conference 1992*, pages 470–479.