Comparing learning approaches to language learning. There is more to it than 'bias'

Véronique Hoste VERONIQUE.HOSTE@UA.AC.BE
Walter Daelemans WALTER.DAELEMANS@UA.AC.BE
CNTS-Language Technology Group, University of Antwerp, Universiteitsplein 1, 2610 Wilrijk, Belgium

Abstract

On the basis of results on different Natural Language Processing (NLP) tasks we show that when following current practice in comparing learning methods, we cannot reliably conclude much about their suitability for a given task. In an empirical study of the behavior of representatives of two machine learning paradigms, viz. lazy learning and rule induction we show that the initial differences between learning techniques are easily overruled when taking into account factors such as feature selection, algorithm parameter optimization, sample selection and their interaction. We propose genetic algorithms as an elegant method to overcome this costly optimization.

1. Introduction

A central question in machine learning research, and more specifically in machine learning of language research is to determine which are the learning algorithms best suited for a given task. Given the 'no free lunch' (Wolpert & Macready, 1995) theorem, this suitability has to be determined experimentally, most often in comparative experiments. In most comparative machine learning experiments, two or more algorithms are compared for a fixed sample selection, feature selection, feature representation, and (default) algorithm parameter setting over a number of trials (cross-validation), and if the measured differences are statistically significant, conclusions are drawn about which algorithm is better suited and why (mostly in terms of algorithm bias). Sometimes different sample sizes are used to provide a learning curve, and sometimes parameters of (some of the) algorithms are optimized on training data, but this is exceptional more than common practice. Many empirical findings, though illustrative, are observations on experiments in which one or two variables are alternated, but in which no overall optimization is undertaken (Mooney (1996), Lee and Ng (2002) and others).

In this paper, we hypothesize that there is a high risk that other areas in the experimental space may lead to radically different results and conclusions. We propose genetic search as an elegant method to overcome the computationally expensive optimization. We test our hypothesis on different types of NLP datasets, including word sense disambiguation, prediction of diminutive suffixes, part of speech tagging, coreference resolution and on 5 UCI benchmark data sets. We experiment with two machine learning methods and discuss some methodological issues involved in running a comparative machine learning (of language) experiment. We will empirically show that changing any of the architectural variables (such as algorithm parameters, information sources, sample selection) can have great effects on the performance of a learning method.

The remainder of this paper is organized as follows. Section 2 presents the two machine learning packages which we used in our experiments and discusses different factors which can influence a machine learning (of language) experiment. In Section 3, we continue with a discussion of the feature selection, the parameter optimization and sample selection experiments. Section 4 reports on the use of a genetic algorithm for joint optimization. We conclude with some general observations in Section 5.

2. A lazy and an eager learner

We experimented with two machine learning techniques: the lazy learning implementation TIMBL (Daelemans et al., 2002) and the eager rule induction method RIPPER (Cohen, 1995). The learning biases of these two approaches provide extremes in the *eagerness* dimension in ML (the degree in which a learning algorithm abstracts from the training data in forming a hypothesis). The motivation for the choice of a lazy or an eager learning bias may be understandability of learned models or abstraction from noise (eager) or the possibility of learning from low-frequency or untypical data points (lazy) to name but a few. But comparing two or more algorithms on a specific task is complex. In Figure 1, a characterization of this complexity is given.

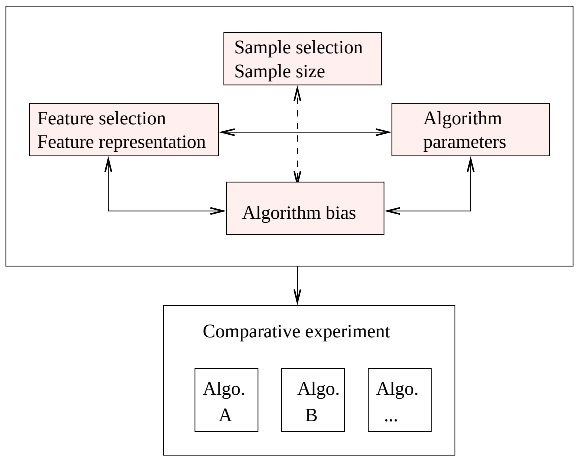


Figure 1. Graphical representation of the aspects influencing a (comparative) machine learning experiment. The filled lines and the dashed line represent the experiments reported in this paper. The dashed line also refers to previous research on sample size of Banko and Brill (2001).

Apart from the algorithm bias, many other factors potentially play a role in the outcome of a machine learning experiment. One of these factors is **the data set**, viz. the sample selection and its size (Banko & Brill, 2001). Also the selection of high-quality training instances has an important effect on predictive accuracy (see for example Zhang (1992) and Skalak (1993) for work on instance and prototype selection). Furthermore, class imbalances in the selected data set can also affect classification results. Another influential factor are the **information sources** used: the **features selected** for prediction, and **their representation** (e.g. binary, numeric or nominal). The presence of irrelevant features can considerably slow the learning rate and have a negative effect on classification results. Furthermore, most learning algorithms have a number of **algorithm parameters** which can be tuned. These factors also interact: a feature selection which is optimal with default parameter settings is not necessarily optimal when changing the algorithm parameters. The optimal algorithm parameters for a skewed data set will not necessarily be optimal when changing the class distribution in the data.

In the following section, we will show that performance differences due to algorithm parameter optimization, feature selection, and sample selection can easily overwhelm the performance differences reported between algorithms in comparative experiments. Due to the large number of experiments performed, we will mainly discuss the observed tendencies which hold for all data sets using one type of NLP task, namely coreference resolution, as test case task. For the GA experiments we report the results on different types of data sets.

3. The effect of optimization

3.1. Coreference resolution as test case task

Coreference can be considered as the act of using a referring expression to point to some discourse entity. Written and spoken texts contain a large number of coreferential relations and a good text understanding largely depends on the correct resolution of these relations. In the following sentences, for example, it is the task of coreference resolution to link "they" to "The kidnappers".

> In 1983 Alfred Heineken and his driver were kidnapped. **The kidnappers** asked a ransom of 43 million guilders. A modest sum, **they** thought.

Machine learning approaches, especially supervised ones, have become increasingly popular for this problem: the C4.5 decision tree learner (Quinlan, 1993) as used by McCarthy (1996) and Soon et al. (2001), the RIPPER rule learner (Cohen, 1995) as in Ng and Cardie (2002) or a memory-based learner (Daelemans et al., 2002) as in Hoste (2005).

For the experiments, we selected all noun phrases in the English MUC-6 (MUC-6, 1995) and MUC-7 (MUC-7, 1998) corpora and the Dutch KNACK-2002 corpus (Hoste, 2005). We selected positive and negative instances for the training data. Positive instances were made by combining each anaphor with each preceding element in the coreference chain (a set of noun phrases referring to the same discourse entity). The negative instances were built by combining each anaphor with each preceding NP which was not part of any coreference chain and by combining each anaphor with each preceding NP which was part of another coreference chain. This resulted in a highly skewed data set. For example, out of the 171,081 training instances in the MUC-6 data merely 6.6% were positive ones. Besides merging all NPs into one single train and test set, we also built 3 smaller datasets, each specialized in one NP type (pronouns, proper nouns, common nouns).

For our coreference resolution system, we used a combination of positional features (features indicating the number of sentences/NPs between the anaphor and its possible antecedent), morphological and lexical features (such as features which indicate whether a given anaphor, its candidate antecedent or both are pronouns, proper nouns, demonstrative or definite NPs), syntactic features which inform on the syntactic function of the anaphor and its candidate antecedent and check for syntactic parallelism, string-matching features which look for complete and partial matches and finally several semantic features. For a detailed overview of the features, we refer to Hoste (2005).

The validation experiments were performed using tenfold cross-validation on the available training data. In order to have an idea of the performance on the minority class, we evaluated the results of our experiments in terms of precision, recall and F_β. Coreference resolution was selected as test case task for the experiments, since it implies a typical language learning task with many exceptional and low-frequency cases. Furthermore, as discussed earlier coreference resolution data sets are also highly skewed and consist of instances with some informative features and many uninformative ones (see for example Soon et al. (2001)).

3.2. Searching the feature space

Although the search for disambiguating features is central in the machine learning research for NLP tasks, there is no general practice to also consider the complex interaction between all these information sources. For our experiments, we used two automated techniques for the selection of the relevant features, viz. backward elimination (John et al., 1994) and bidirectional hill-climbing (Caruana & Freitag, 1994). Table 1 gives the results of these experiments for TIMBL and RIPPER on the task of coreference resolution (MUC-7 data set). It shows that (i) the algorithm-comparing differences can be overruled by the algorithm-internal performance differences and that (ii) especially TIMBL can benefit from feature selection which is mainly due to the embedded feature selection in the construction of the rules in RIPPER and the fact that TIMBL does not take into account dependencies between features.

With respect to the selected features, no general conclusions could be drawn. Per data set and per selection procedure, a different feature set is selected by each learner, which implies that the optimal feature selection has to be determined experimentally for each single data set.

3.3. The effect of parameter optimization

Another factor which can have great effects on classifier performance is the choice of algorithm parameter settings. Although both algorithms provide sensible default settings, it is by no means certain that they are the optimal settings for our task. Therefore, we exhaustively varied the algorithm parameter settings for each classifier. For TIMBL, the following parameters were varied: the similarity metric (overlap or modified value difference metric (MVDM)), feature weighting (no weighting, information gain weighting, gain ratio weighting, ...), neighbor weighing (majority voting and different distance weighting schemes), and the k parameter which controls the number of nearest neighbors. For RIPPER, there was an optimization of the class ordering parameter (+freq, -freq or mdl), the two-valued negative tests parameter (-!n or nothing), the hypothesis simplification parameter (0.5, 1 or 1.5), the example coverage parameter (1, 2, 3 or 4), the parameter expressing the number of optimization passes (0, 1 or 2) and the loss ratio parameter (0.5, 1, 1.5). Figure 2 displays the $F_{\beta=1}$ results of all algorithm parameter optimization experiments for the MUC-6 coreference resolution data sets. Per algorithm and per data set, the best and worst scores are displayed, as well as the averages and deviations. The long vertical lines in Figure 2 reveal a lot of variation in the $F_{\beta=1}$ results when varying the algorithm parameters, although the boxes which are mostly located in the upper area indicate that the badly performing parameter combinations are in the minority. A good parameter combination is crucial. In the MUC-6 common nouns data set, for example, RIPPER yields an $F_{\beta=1}$ score of 17.65% for the combination of the '-freq' ordering method and the '0.5' loss ratio value. Combining this below zero loss ratio value with the '+freq' or 'mdl' ordering methods, however, leads to top $F_{\beta=1}$ scores on the validation material. Similar observations can be made for TIMBL.

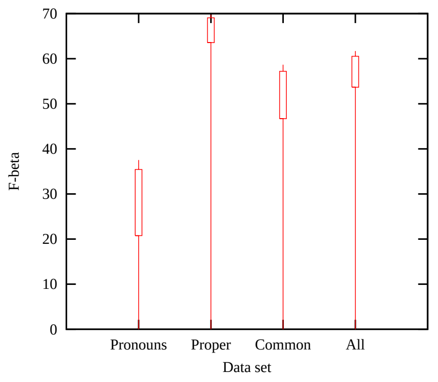
Overall, we observed that parameter optimization leads to large performance increases for both learners. Furthermore, we observed that parameters cannot be generalized. The optimal settings merely reveal some tendencies.

3.4. The effect of sample selection

NLP data sets often reveal large class imbalances. In the KNACK-2002 coreference resolution cross-validation data, for example, merely 6.3% of the instances is classified as positive. Learning performance can be hindered when learning from these data sets where the minority class is underrepresented. A cen-

Table 1. Results of TIMBL and RIPPER on the task of coreference resolution (MUC-7 data sets) after backward selection and bidirectional hill-climbing.

		TIMBL			RIPPER		
		Prec.	Rec.	$F_{\beta=1}$	Prec.	Rec.	$F_{\beta=1}$
All	default	51.57	46.09	48.68	77.51	36.21	49.36
	backward	73.98	41.26	52.98	77.49	40.34	53.06
	bi.hill.	75.39	42.08	**54.01**	77.99	40.52	**53.33**
Pronouns	default	42.31	36.60	39.25	59.50	22.70	32.86
	backward	46.86	40.89	**43.67**	59.34	26.43	36.57
	bi.hill.	61.55	25.51	36.07	60.74	30.94	**41.00**
Proper nouns	default	62.36	56.87	59.49	84.58	52.56	64.83
	backward	73.92	55.54	63.43	87.08	55.22	**67.59**
	bi.hill.	85.18	54.88	**66.75**	88.20	51.72	65.21
Common Nouns	default	43.06	39.17	41.03	74.56	36.76	49.24
	backward	52.02	39.28	44.76	76.05	40.41	**52.78**
	bi.hill.	78.83	38.72	**51.93**	76.77	39.70	52.33



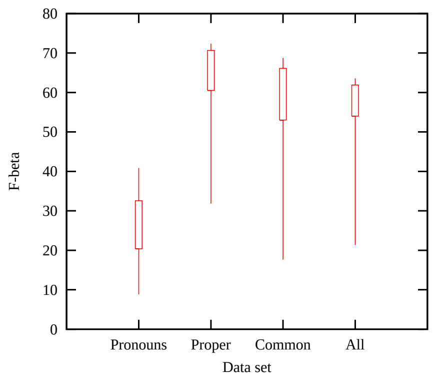


Figure 2. Results of TIMBL (left) and RIPPER (right) over all parameter settings for the MUC-6 coreference resolution data sets. The graphs show the difference between the performance obtained with the best and worst parameter settings per data set. The boxes in the graphs represent averages and deviations.

tral question in the discussion on data sets with an imbalanced class distribution is in what proportion the classes should be represented in the training data. One can argue that the natural class distribution should be used for training, even if it is highly imbalanced, since a model can then be built which fits a similar imbalanced class distribution in the test set. Others believe that the training set should contain an increased number of minority class examples. In the machine learning literature, there have been several proposals (see Japkowicz and Stephen (2002)) for adjusting the number of majority class and minority class examples. Methods include resizing training data sets or sampling, adjusting misclassification costs, learning from the minority class, adjusting the weights of the examples, etc.

In order to investigate the effect of class distribution on classifier performance, we compared the performance of the classifiers on a variety of class distributions. We investigated the effect of random down-sampling and down-sampling of the true negatives for both TIMBL and RIPPER. This was done by gradually downsizing the number of negatives instances in slices of 10% until there was an equal number of positive and negative training instances. These experiments reveal the same tendencies for the different data sets. As exemplified in Figure 3 for the MUC-6 coreference resolution data sets, we can observe that TIMBL and RIPPER behave differently. RIPPER is more sensitive to the skewedness of the classes and down-sampling is beneficial for the RIPPER results. Furthermore, down-sampling only starts being harmful at a high down-sampling level. TIMBL has shown this tendency only on the "Pronouns" data set. But no down-sampling level leads to the best performance over all data sets.

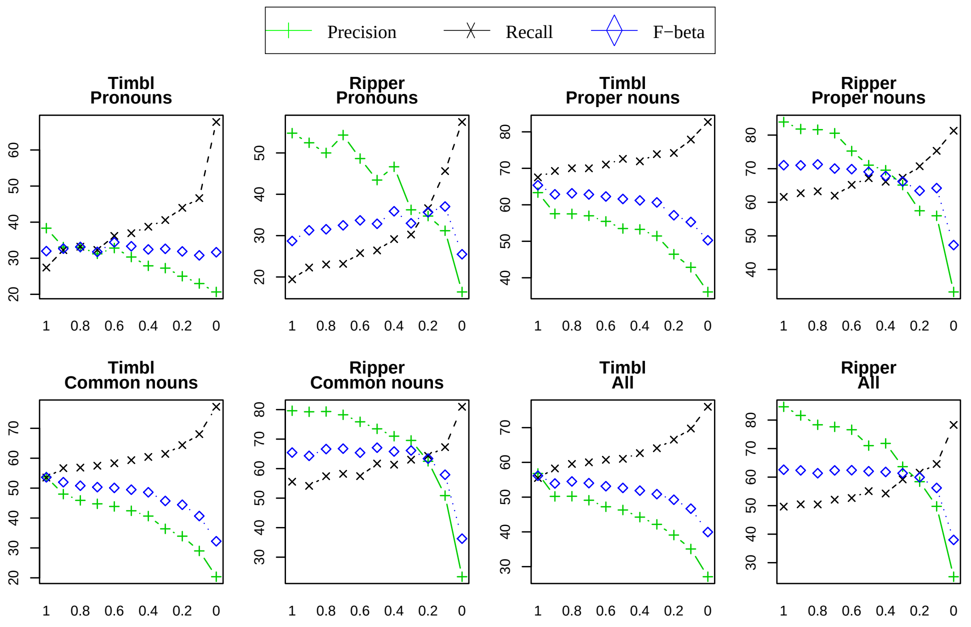
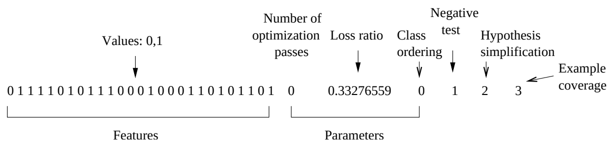


Figure 3. Cross-validation results in terms of precision, recall and $F_{\beta=1}$ after application of TIMBL and RIPPER on the MUC-6 coreference resolution data with a randomly down-sampled majority class. The test partitions keep their initial class distribution.

4. Genetic algorithms for joint optimization

In a final optimization step we explored the interaction between the previously mentioned factors. Joint feature selection, sample selection and parameter optimization is essentially an optimization problem which involves searching the space of all possible feature subsets, sample subsets and parameter settings to identify the combination that is optimal or near-optimal. Given the combinatorially explosive character of this type of joint optimization, we have chosen for genetic algorithms (GA, e.g. Goldberg (1989) and Mitchell (1996)) as a computationally feasible way to achieve this. One of the advantages of genetic algorithms in contrast to local search methods such as hill-climbing, gradient based and simulated annealing methods is that they explore different areas of the search space in parallel, which might be a reasonable strategy for problems with a large number of parameters and features. GAs contain at any time a population of candidate solutions to the optimization problem to be solved. For the experiments, we used a generational genetic algorithm with the evaluations distributed over a cluster of computers using the Sun Grid Engine queuing system. The following GA parameters were used and kept constant: maximal number of generations=30, population size=10, uniform crossover (crossover rate=0.9), tournament selection (selection size=2), discrete (mutation rate=0.2) and Gaussian mutation (k and loss ratio). $F_{\beta=1}$ was used as fitness function. We are aware that the optimization problem we are trying to solve with a genetic algorithm also applies to the GA itself.



In the experiments, the individuals were represented as bit strings. Each individual contains particular values for all algorithm settings and for the selection of the features. For example, for RIPPER, the features are encoded as binary alleles. At the end of the chromosome, the different algorithm parameters are represented. Through the variation of the loss ratio parameter, which controls the relative weight of precision versus recall, a down-sampling effect can be obtained.

The GA optimization experiments confirm the tendencies observed in the optimization experiments in the previous section. As exemplified for the KNACK-2002 coreference resolution data in Figure 4, we could observe for all data sets that the performance differences inside one single learning method can be much larger than the method-comparing performance differences. In their default representation, for example,

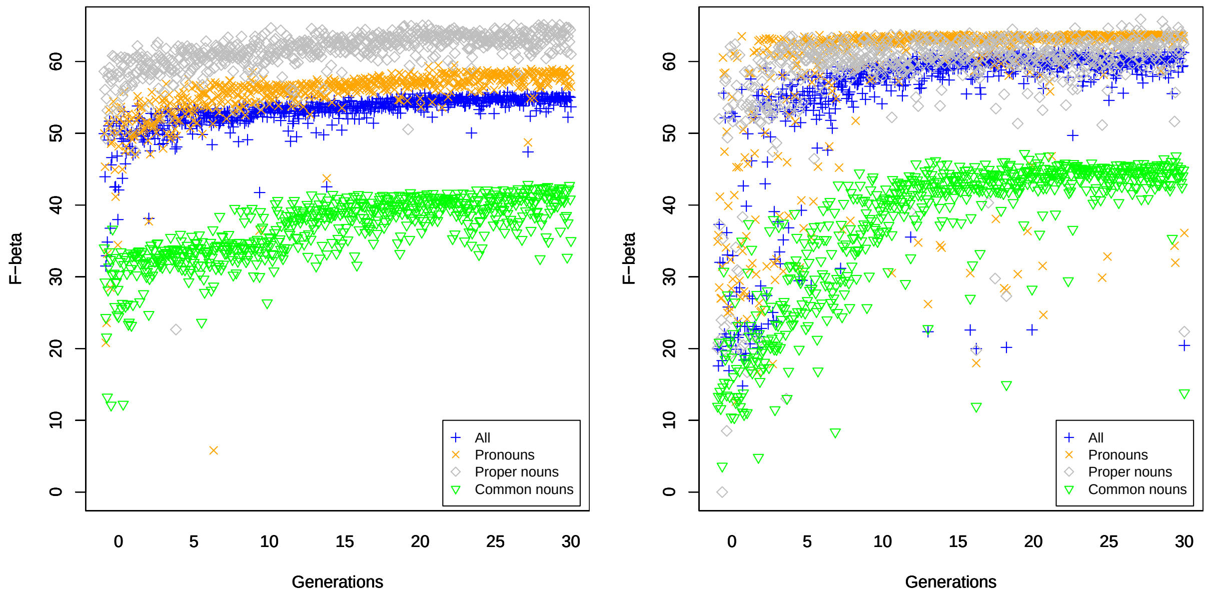


Figure 4. Results of TIMBL (left) and RIPPER (right) for the KNACK-2002 data sets during GA optimization.

TIMBL and RIPPER yield a 46.8% and a 46.5% $F_{\beta=1}$ score, respectively. Optimization leads to a large performance improvement for both learners and to a reversed supremacy: 55.7% for TIMBL and 61.7% for RIPPER. In conclusion, we can state that we cannot draw conclusions of one classifier being better on a particular task than another classifier, when only taking into account default settings or limited optimization.

These observations, however, are not limited to the task of coreference resolution. Table 2 clearly exemplifies the usefulness of parameter optimization for the task of word sense disambiguation (WSD). WSD is a natural language processing task in which a word with more than one sense has to be disambiguated by using information from the context in which the word occurs. E.g. *knight* can refer to a chess piece or a medieval character. Word sense disambiguation is a crucial component in applications such as machine translation (depending on the sense, the English *knight* will be translated into a French *chevalier* or *cavalier*), question answering, information retrieval, etc. Over the last years, three senseval [1] have been organised to evaluate the strenghts and weaknesses of WSD methods. Machine learning methods such as decision list learning and memory-based learning have been shown to outperform hand-crafting approaches in these comparisons. Table 2 shows the results on the English lexical sample data in the Senseval-3 competition. It clearly shows the usefulness of GA optimization. In (Hoste et al., 2002; Daelemans & Hoste, 2002; Daelemans et al., 2003) we came to similar conclusions on the tasks of diminutive prediction and part of speech tagging.

[1]http://www.senseval.org

Furthermore, this effect of optimization is not limited to natural language processing datasets. We performed experiments on 5 UCI benchmark datasets[2]: “database for fitting contact lenses” (24 instances), “contraceptive method choice” (1473 instances), “breast-cancer-wisconsin” (699 instances), “car evaluation data base” (1728 instances) and “postoperative patient data” (90 instances). And we came to similar conclusions as on the NLP data sets, as shown in Table 3.

These effects explain why in the machine learning of natural language literature, so many results and interpretations about the superiority of one algorithm over the other are contradictory. We show that there is a high risk that other areas in the experimental space may lead to radically different results and conclusions.

[2]http://www/ics/uci/edu/˜mlearn/MLRepository.html

	TRAINING SET		TEST SET			TRAINING SET		TEST SET	
LEMMA/POS	DEF	OPT	DEF	OPT	LEMMA/POS	DEF	OPT	DEF	OPT
provide/v	84.56	94.85	88.40	**92.75**	rule/n	75.44	91.23	50.00	**60.00**
eat/v	79.04	89.22	78.16	**91.95**	image/n	49.00	62.69	48.64	**56.75**
remain/v	85.40	95.62	82.85	**88.57**	paper/n	37.95	54.46	38.46	**55.55**
arm/n	88.67	93.20	84.21	**84.96**	produce/v	50.54	65.22	53.19	**55.31**
plan/v	67.93	78.48	75.00	**83.33**	suspend/v	46.34	59.35	34.37	**51.56**
add/v	73.95	82.38	79.54	**82.57**	argument/n	42.04	57.58	43.24	**51.35**
degree/n	64.56	78.38	71.09	**82.03**	difficulty/n	35.48	58.06	34.78	**39.13**
hot/a	68.67	78.00	76.74	**81.39**	performance/n	38.21	52.85	28.73	**39.08**
watch/v	85.71	89.80	78.43	**80.39**	use/v	80.77	88.46	**78.57**	**78.57**
smell/v	70.41	85.27	74.54	**78.18**	hear/v	64.52	74.19	**53.12**	**53.12**
bank/n	61.36	79.22	59.84	**78.03**	win/v	50.65	68.83	**48.71**	**48.71**
expect/v	64.93	77.92	73.07	**76.92**	different/a	54.81	65.27	**46.00**	**46.00**
talk/v	77.37	83.21	73.97	**75.34**	miss/v	40.00	68.89	**43.33**	**43.33**
appear/v	79.24	87.17	71.42	**75.18**	solid/a	9.80	31.78	**27.58**	**27.58**
decide/v	72.95	86.89	70.96	**74.19**	receive/v	75.00	80.77	**92.59**	88.88
wash/v	32.26	62.90	52.94	**73.52**	organization/n	67.66	77.51	69.64	**73.21**
mean/v	84.81	91.14	**77.50**	75.00	audience/n	73.90	85.29	**76.00**	74.00
party/n	61.82	71.96	65.51	**72.41**	operate/v	72.73	84.85	**66.66**	55.55
interest/n	63.28	70.36	59.13	**72.04**	write/v	64.29	71.43	**56.52**	43.47
express/v	48.62	72.48	45.45	**70.90**	play/v	48.42	64.21	**51.92**	42.30
sort/v	61.09	78.60	66.66	**70.83**	difference/n	57.14	68.51	**47.36**	46.49
treat/v	37.84	55.86	**40.35**	38.59	judgment/n	35.64	60.40	**40.62**	34.37
note/v	56.15	69.23	61.19	**68.65**	atmosphere/n	47.42	60.20	51.85	**70.37**
disc/n	54.03	69.19	52.00	**66.00**	encounter/v	51.94	65.89	58.46	**60.00**
climb/v	63.48	78.26	59.70	**64.17**	important/a	72.08	82.23	42.10	**47.36**
shelter/n	66.14	74.02	54.08	**63.26**	activate/v	70.40	80.27	64.91	80.70
simple/a	43.55	58.52	44.44	**61.11**	source/n	34.06	52.90	46.87	59.37
ask/v	49.80	62.06	60.30	**61.06**	OVERALL SCORE				
begin/v	53.41	63.07	53.16	**60.75**	FINE-GR.	59.82	71.28	60.80	**67.40**
lose/v	44.78	62.69	36.11	**52.77**	COARSE-GR.	/	/	/	**74.00**

Table 2. Classification accuracies for all lemmas in the English lexical sample task. The first column presents the words to be disambiguated, together with their part-of-speech. The second and third column present the default results and the optimized results of TIMBL on the validation data, whereas the last two columns contain the default and optimized scores on the official Senseval-3 test data.

5. Concluding remarks

In this paper, we showed that many factors can affect the success of a classifier, such as the specific 'bias' from the classifier, the choice of algorithm parameters, the selection of information sources, the sample selection and the interaction between these factors. We also showed that optimization can lead to radically different results, causing much larger classifier-internal variations than classifier-comparing variations. These results call into question the usefulness of the numerous classifier comparison studies in the literature. On the other hand, significant performance increases can be obtained this way. We conclude that in general, the more effort is put in optimization, through feature selection, parameter optimization, sample selection and their joint optimization, the more reliable the results and the comparison will be.

References

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (pp. 26–33).

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. *Proceedings of the International Conference on Machine Learning* (pp. 28–36).

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the 12th International Conference on Machine Learning* (pp. 115–123).

Daelemans, W., & Hoste, V. (2002). Evaluation of machine learning methods for natural language processing tasks. *Proceedings of the Third International Conference on Language Resources and Evaluation* (pp. 755–760).

Daelemans, W., Hoste, V., De Meulder, F., & Naudts,

Table 3. Classification accuracies of TIMBL and RIPPER on 5 UCI benchmark datasets.

Dataset		TIMBL	RIPPER
Database for fitting contact lenses	Def.	75.0	79.2
	GA opt.	87.5	87.5
Contraceptive method choice	Def.	48.5	46.8
	GA opt.	54.8	49.8
Breast-cancer-wisconsin	Def.	95.7	93.7
	GA opt.	97.6	95.7
Car evaluation database	Def.	94.0	87.0
	GA opt.	96.9	98.4
Postoperative patient data	Def.	55.6	71.1
	GA opt.	71.1	71.1

B. (2003). Combined optimization of feature selection and algorithm parameter interaction in machine learning of language. *Proceedings of the 14th European Conference on Machine Learning* (pp. 84–95).

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2002). *Timbl: Tilburg memory-based learner, version 4.3, reference guide* (Technical Report ILK Technical Report - ILK 02-10). Tilburg University.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison Wesley.

Hoste, V. (2005). *Optimization issues in machine learning of coreference resolution.* Doctoral dissertation, Antwerp University.

Hoste, V., Hendrickx, I., Daelemans, W., & van den Bosch, A. (2002). Parameter optimization for machine-learning of word sense disambiguation. *Natural Language Engineering, Special Issue on Word Sense Disambiguation Systems*, *8*, 311–325.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, *6*, 429–450.

John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *International Conference on Machine Learning* (pp. 121–129).

Lee, Y., & Ng, H. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (pp. 41–48).

McCarthy, J. (1996). *A trainable approach to coreference resolution for information extraction.* Doctoral dissertation, Department of Computer Science, University of Massachusetts, Amherst MA.

Mitchell, M. (1996). *An introduction to genetic algorithms.* MIT Press.

Mooney, R. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In E. Brill and K. Church (Eds.), *Proceedings of the conference on empirical methods in natural language processing*, 82–91.

MUC-6 (1995). Coreference task definition. version 2.3. *Proceedings of the Sixth Message Understanding Conference* (pp. 335–344).

MUC-7 (1998). Muc-7 coreference task definition. version 3.0. *Proceedings of the Seventh Message Understanding Conference.*

Ng, V., & Cardie, C. (2002). Combining sample selection and error-driven pruning for machine learning of coreference rules. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (pp. 55–62).

Quinlan, J. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, CA.

Skalak, D. B. (1993). Using a genetic algorithm to learn prototypes for case retrieval and classification. *Proceedings of the AAAI-93 Case-Based Reasoning Workshop* (pp. 64–69).

Soon, W., Ng, H., & Lim, D. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, *27*, 521–544.

Wolpert, D., & Macready, W. (1995). *No free lunch theorems for search* (Technical Report SFI-TR-95-02-010). Santa Fe Institute, Santa Fe, NM.

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