Optimizing to Arbitrary NLP Metrics using Ensemble Selection

Art Munson, Claire Cardie, Rich Caruana
Department of Computer Science
Cornell University
Ithaca, NY 14850
{mmunson, cardie, caruana}@cs.cornell.edu

Abstract

While there have been many successful applications of machine learning methods to tasks in NLP, learning algorithms are not typically designed to optimize NLP performance metrics. This paper evaluates an ensemble selection framework designed to optimize arbitrary metrics and automate the process of algorithm selection and parameter tuning. We report the results of experiments that instantiate the framework for three NLP tasks, using six learning algorithms, a wide variety of parameterizations, and 15 performance metrics. Based on our results, we make recommendations for subsequent machine-learning-based research for natural language learning.

1 Introduction

Among the most successful natural language learning techniques for a wide variety of linguistic phenomena are supervised inductive learning algorithms for classification. Because of their capabilities for accurate, robust, and efficient linguistic knowledge acquisition, they have been employed in many natural language processing (NLP) tasks.

Unfortunately, supervised classification algorithms are typically designed to optimize accuracy (*e.g.* decision trees) or mean squared error (*e.g.* neural networks). For many NLP tasks, however, these standard performance measures are inappropriate. For example, NLP data can be highly skewed in its distribution of positive and negative examples. In these situations, another metric (perhaps F-measure or a task-specific measure) that focuses on the performance of the minority cases is more appropriate. Indeed, as the NLP field matures more consideration will be given to evaluating the performance of NLP components in context (*e.g.* Is the system easy to use by end users? Does the component respect user preferences? How well does the entire system solve the specific problem?), leading to new and complicated metrics. Optimizing machine learning algorithms to arbitrary performance metrics, however, is not easily done.

To exacerbate matters, the metric of interest might change depending on how the natural language learning (NLL) component is employed. Some applications might need components with high recall, for example; others, high precision or high F-measure or low root mean squared error. To obtain good results w.r.t. the new metric, however, a different parameterization or different algorithm altogether might be called for, requiring re-training the classifier(s) from scratch.

Caruana *et al.* (2004) have recently proposed **ensemble selection** as a technique for building an ensemble of classifiers that is optimized to an arbitrary performance metric. The approach trains a large number of classifiers using multiple algorithms and parameter settings, with the idea that at least some of the classifiers will perform well on any given performance measure. The best set of classifiers, w.r.t. the target metric, is then greedily selected. (Selecting a set of size 1 is equivalent to parameter and algorithm tuning.) Like other ensemble learning methods (*e.g.* bagging (Breiman, 1996) and boosting (Freund and Schapire, 1996)), ensemble selection has been shown to exhibit reliably better performance than any of the contributing classifiers for a number of learning tasks.

In addition, ensemble selection provides an ancil-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 539–546, Vancouver, October 2005. ©2005 Association for Computational Linguistics

lary benefit: no human expertise is required in selecting an appropriate machine learning algorithm or in choosing suitable parameter settings to get good performance. This is particularly attractive for the NLP community where researchers often rely on the same one or two algorithms and use default parameter settings for simplicity. Ensemble selection is a tool usable by non-experts to find good classifiers optimized to task-specific metrics.

This paper evaluates the utility of the ensemble selection framework for NLL. We use three NLP tasks for the empirical evaluation: noun phrase coreference resolution and two problems from sentiment analysis — identifying private state frames and the hierarchy among them. The evaluation employs six learning algorithms, a wide selection of parameterizations, 8 standard metrics, and 7 task-specific metrics. Because ensemble selection subsumes parameter and algorithm selection, we also measure the impact of parameter and algorithm tuning.

Perhaps not surprisingly, we find first that no one algorithm or parameter configuration performs the best across all tasks or across all metrics. In addition, an algorithm's "tuned" performance (*i.e.* the performance after tuning parameter settings) almost universally matches or exceeds the algorithm's default performance (*i.e.* when using default parameter settings). Out of 154 total cases, the tuned classifier outperforms the default classifier 114 times, matches performance 28 times, and underperforms 12 times. Together, these results indicate the importance of algorithm and parameter selection for comparative empirical NLL studies. In particular, our results show the danger of relying on the same one or two algorithms for all tasks. These results cast doubt on conclusions regarding differences in algorithm performance for NLL experiments that give inadequate attention to parameter selection.

The results of our experiments that use ensemble selection to optimize the ensemble to arbitrary metrics are mixed. We see reliable improvements in performance across almost all of the metrics for only two of the three tasks; for the other data set, ensemble selection tends to hurt performance (although losses are very small). Perhaps more importantly for our purposes, we find that ensemble selection provides small, but consistent gains in performance when considering only the more complex, task-specific metrics — metrics that learning algorithms would find difficult to optimize.

The rest of the paper is organized as follows. Section 2 describes the general learning framework and provides an overview of ensemble selection. We present the particular instantiation of the framework employed in our experiments in Section 3. Section 4 describes the three NLP tasks. Experimental results are given in Section 5. Related work and conclusions follow (sections 6 and 7).

2 Ensemble Selection Framework

2.1 Terminology

We use the term **model** to refer to a classifier produced by some learning algorithm using some particular set of parameters. A model's **configuration** is simply the algorithm and parameter settings used to create the classifier. A **model family** is the set of models made by varying the parameters for one machine learning algorithm. Finally, a **model library** is a collection of models trained for a given task.

2.2 Framework

Abstractly, the framework is the following:

1. Select a variety of learning algorithms.
2. For each algorithm, choose a wide range of settings for the algorithm's parameters.
3. Divide data into training, tuning, and test sets.
4. Build model library.
5. Select target metrics appropriate to problem.
6. Tune parameter settings and/or run ensemble selection algorithm for target metrics.

Building the model library consists of (a) using the training data to train models for all the learning algorithms under all desired combinations of parameter settings, and (b) applying each model to the tuning and test set instances and storing the predictions for use in step (6). Note that models are placed in the library regardless of performance, even though some models have very bad performance. Intuitively, this is because there is no way to know *a priori* which models will perform well on a given metric. Note that producing the base models is fully automatic and requires no expert tuning.

Parameter Tuning: The goal of parameter tuning is to identify the best model for the task according to each target metric. Parameter tuning is handled in the standard way: for each metric, we select the model from the model library with the highest performance on the tuning data and report its performance on the test data.

Ensemble Selection Algorithm: The ensemble selection algorithm (Caruana *et al.*, 2004) ignores model-specific details by *only using the predictions made by the models:* the ensemble makes predictions by averaging the predictions of its constituents. Advantageously, this only requires that predictions made by different models fall in the same range, and that they can be averaged in a meaningful way. Otherwise, models can take any form, including other ensemble methods (*e.g.* bagging or boosting). Conceptually, ensemble selection builds on top of the models in the library and uses their performance as a starting point from which to improve.

The basic ensemble selection algorithm is:

a. Start with an empty ensemble.
b. Add the model that results in the best performance for the current ensemble with respect to the tuning data and the target metric.
c. Repeat (b) for a large, fixed number of iterations.
d. The final ensemble is the ensemble from the best performing iteration on the tuning data for the target metric.

To prevent the algorithm from overfitting the tuning data we use two enhancements given by Caruana *et al.* (2004). First, in step (b) the same model can be selected multiple times (*i.e.* selection with replacement). Second, the ensemble is initialized with the top N models (again, with respect to the target metric on the tuning data). N is picked automatically such that removing or adding a model decreases performance on the tuning data.[1]

The main advantage to this framework is its reusability. After an instantiation of the framework exists, it is straightforward to apply it to multiple NLL tasks and to add additional metrics. Steps (1) and (2) only need to be done once, regardless of the number of tasks and metrics explored. Steps (3)-(5) need only be done once per NLL task. Importantly, the model library is created once for each task (*i.e.* each model configuration is only trained once) regardless of the number (or addition) of performance metrics. Finally, finding a classifier or ensemble optimized to a new metric (step (6)) does not require re-building the model library and is very fast compared to training the classifiers—it only requires averaging the stored predictions. For example, training the model library for our smallest data set took multiple days; ensemble selection built optimized ensembles for each metric in a few minutes.

3 Framework Instantiation

In this section we describe our instantiation of the ensemble selection framework.

Algorithms: We use bagged decision trees (Breiman, 1996), boosted decision stumps (Freund and Schapire, 1996), k-nearest neighbor, a rule learner, and support vector machines (SVM's). We use the following implementations of these algorithms, respectively: `IND` decision tree package (Buntine, 1993); `WEKA` toolkit (Witten and Frank, 2000); `TiMBL` (Daelemans *et al.*, 2000); `RIPPER` (Cohen, 1995); and SVM^{light} (Joachims, 1999). Additionally, we use logistic regression (LR) for coreference resolution because an implementation using the `MALLET` (McCallum, 2002) toolkit was readily available for the task. The predictions from all algorithms are scaled to the range $[0, 1]$ with values close to 1 indicating positive predictions and values close to 0 indicating negative predictions.[2]

Parameter Settings: Table 1 lists the parameter settings we vary for each algorithm. Additional domain-specific parameters are also varied for coreference resolution models (see Section 4.1). The model libraries contain models corresponding to the cross product of the various parameter settings for a given algorithm.

Standard Performance Metrics: We evaluate the framework with 8 metrics: accuracy (ACC), average precision (APR), break even point (BEP), F-measure (F1), mean cross entropy (MXE), root mean squared error (RMS), area under the ROC curve (ROC), and SAR. Caruana *et al.* (2004) define SAR as $SAR = \frac{ACC+ROC+(1-RMS)}{3}$. We also evaluate the effects of model selection with task-specific metrics. These are described in Section 4. Our F-measure places equal emphasis on precision

[1] We also experimented with the bagging improvement described by Caruana *et al.* (2004). In our experiments using bagging hurt the performance of ensemble selection.

[2] We follow Caruana *et al.* (2004) in using Platt (2000) scaling to convert the SVM predictions from the range $(-\infty, \infty)$ to the required $[0, 1]$ by fitting them to a sigmoid.

Algorithm	Parameter	Values
Bagged Trees †	tree type	bayes, **c4**, cart, cart0, id3, mml, smml
	# bags	1, 5, 10, **25**
Boosted Stumps	# iterations	2, 4, 8, …, **256**, …1024, 2048
LR ‡	gaussian gamma	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, **10**, 50
K-NN	k	**1**, 3, 5
	search algorithm	**ib1**, igtree
	similarity metric	**overlap**, modified value difference
	feature weighting	**gain ratio**, information gain, chi-squared, shared variance
Rule Learner	class learning order	unordered, **pos first**, neg first, heuristic determined order
	loss ratio	0.5, **1**, 1.5, 2, 3, 4
SVM	margin tradeoff*	$10^{-7}, 10^{-6}, \ldots, \mathbf{10^{-2}}, \mathbf{10^{-1}}, \ldots, 10^{2}$
	kernel	**linear**, rbf
	rbf gamma parm	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2

Table 1: Summary of model configurations used in experiments. The default settings for each algorithm are in bold. † Bagged trees are not used for identifying PSF's since the IND package does not support features with more than 255 values. Also, for coreference resolution the number of bags is not varied and is always 25. ‡ LR is only used for coreference resolution. * SVM^{light} determines the default margin tradeoff based on data properties. We calculate this value for each data set and use the closest setting among our configurations.

and recall (*i.e.* $\beta = 1$). Note that precision and recall are calculated *with respect to the positive class*.

Ensemble Selection: For the sentiment analysis tasks, ensemble selection iterates 150 times; for the coreference task, the algorithm iterates 200 times. This should be enough iterations, given that the model libraries contain 202, 173, and 338 models. Because computing the MUC-F1 and BCUBED metrics (see Section 4.1) is expensive, ensemble selection only iterates 50 times for these metrics.

4 Tasks

Because of space issues, we necessarily provide only brief descriptions of each NLL task. Readers are referred to the cited papers to obtain detailed descriptions.

4.1 Noun Phrase Coreference Resolution

The goal for a standard noun phrase coreference resolution system is to identify the noun phrases in a document and determine which of them refer to the same entity. Entities can be people, places, things, *etc*. The resulting partitioning of noun phrases creates reference chains with one chain per entity.

We use the same problem formulation as Soon *et al.* (2001) and Ng and Cardie (2002) — a combination of classification and clustering. Briefly, every noun phrase is paired with all preceding noun phrases, creating multiple pairs. For the training data, the pairs are labeled as coreferent or not. A binary classifier is trained to predict the pair labels. During classification, the predicted labels are used to form clusters. Two noun phrases A and B share a cluster if they are either predicted as coreferent by the classifier or if they are transitively predicted as coreferent through one or more other noun phrases. Instance selection (Soon *et al.*, 2001; Ng, 2004) is used to increase the percentage of positive instances in the training set.[3]

We use the learning features described by Ng and Cardie (2002). All learning algorithms are trained with the full set of features. Additionally, the rule learner, SVM, and LR are also trained with a hand-selected subset of the features that Ng and Cardie (2002) find to outperform the full feature set. Essentially this is an additional parameter to set for the learning task.

Special Metrics: Rather than focusing on performance at the pairwise coreference classification level, performance for this task is typically reported using either the MUC metric (Vilain *et al.*, 1995) or the BCUBED metric (Bagga and Baldwin, 1998). Both of these metrics measure the degree that predicted coreference chains agree with an answer key. In particular they measure the chain-level precision and recall (and the corresponding F-measure). We abbreviate these metrics MUC-F1, and B3F1.

Data Set: For our experiments we use the MUC-6 corpus, which contains 60 documents annotated with coreference information. The training, tuning, and test sets consist of documents 1-20, 21-30, and

[3] `Soon-1` instance selection is used for all algorithms; we also use `soon-2` (Ng, 2004) instance selection for the rule learner.

31-60, respectively.

4.2 Identifying Private State Frames

NLP research has recently started looking at how to detect and understand subjectivity in discourse. A key step in this direction is automatically identifying phrases that express subjectivity. In this setting, subjectivity is defined to include implicit and explicit opinions, internal thoughts and emotions, and bias introduced through second-hand reporting. Phrases expressing any of these are called **private state frames**, which we will abbreviate as PSF.

We build directly on experiments done by Wiebe *et al.* (2003). The task is to learn to identify explicit single-word PSF's in context. One learning instance is created for every word in the corpus. Classification labels each instance as a PSF or not. We use the same features as Wiebe *et al.*

Special Metrics: Because the data is highly skewed (2% positive instances), performance measures that focus on how well the minority class is learned are of primary interest. The F-measure defined in Section 3 is a natural choice. We also evaluate performance using geometric accuracy, defined as $GACC = \sqrt{posacc \times negacc}$ where $posacc$ and $negacc$ are the accuracy with respect to the positive and negative instances (Kubat and Matwin, 1997).

Conceivably, an automatic PSF extractor with high precision and mediocre recall could be used to automate the annotation process. For this reason we measure the performance with an unbalanced F-measure that emphasizes precision. Specifically, we try $\beta = 0.5$ (F0.5) and $\beta = 0.2$ (F0.2).

Data Set: We use 400 documents from the MPQA corpus (2002), a collection of news stories manually annotated with PSF information. The 400 documents are randomly split to get 320 training, 40 tuning, and 40 testing documents.

4.3 Determining PSF Hierarchy

The third task is taken from Breck and Cardie (2004). Explicit PSF's each have a **source** that corresponds to the person or entity expressing the subjectivity. In the presence of second-hand reporting, sources are often nested. This has the effect of filtering subjectivity through a chain of sources.

Given sentences annotated with PSF information (*i.e.* which spans are PSF's), the task is to discover the hierarchy among the PSF's that corresponds to the nesting of their respective sources. From each sentence, multiple instances are created by pairing every PSF with every other PSF in the sentence.[4] Let $(PSF_{parent}, PSF_{target})$ denote one of these instances. The classification task is to decide if PSF_{parent} is the parent of PSF_{target} in the hierarchy and to associate a confidence with that prediction. The complete hierarchy can easily be constructed from the predictions by choosing for each PSF its most confidently predicted parent.

Special Metrics: Breck and Cardie (2004) measure task performance with three metrics. The first is the accuracy of the predictions over the instances. The second is a derivative of a measure used to score dependency parses. Essentially, a sentence's score is the fraction of parent links correctly identified. The score for a set of sentences is the average of the individual sentence scores. We refer to this measure as average sentence accuracy (SENTACC). The third measure is the percentage of sentences whose hierarchical structures are perfectly determined (PERFSENT).

Data Set: We use the same data set and features as Breck and Cardie (2004). The annotated sentences from 469 documents in the MPQA Corpus (MPQA Corpus, 2002) are randomly split into training (80%), tuning (10%), and test (10%) sets.

5 Experiments and Results

We evaluate the potential benefits of the ensemble selection framework with two experiments. The first experiment measures the performance improvement yielded by parameter tuning and finds the best performing algorithm. The second experiment measures the performance improvement from ensemble selection.

Performance improvements are measured in terms of performance **gain**. Let a and b be the measured performances for two models A and B on some metric. A's gain over B is simply $a - b$. A performed worse than B if the gain is negative.[5]

[4] Sentences containing fewer than two PSF's are discarded and not used.

[5] MXE and RMS have inverted scales where the best performance is 0. Gain for these metrics equals $(1 - a) - (1 - b)$ so that positive gains are always good. Similarly, where raw MXE and RMS scores are reported we show $1 - score$.

	Metric	BAG	Parm∆	BST	Parm∆	LR	Parm∆	KNN	Parm∆	RULE	Parm∆	SVM	Parm∆	Avg ∆
coreference	ACC	0.9861	-0.0000	*0.9861*	-0.0001	0.9849	0.0006	0.9724	0.0131	0.9840	**0.0023**	0.9859	-0.0001	0.0026
	APR	*0.5373*	**0.0000**	0.5475	0.0000	0.3195	-0.0004	0.1917	0.2843	0.2491	0.1127	0.5046	0.0323	0.0715
	BEP	*0.6010*	0.0000	0.5577	0.0193	0.3747	-0.0022	0.3243	0.2057	0.3771	0.1862	0.5965	**0.0045**	0.0689
	F1	*0.5231*	**0.0664**	0.3881	0.0000	0.4600	0.0087	0.4105	0.1383	0.4453	0.1407	0.3527	0.0571	0.0685
	MXE	0.9433	**0.0082**	*0.9373*	0.0000	0.5400	0.1828	0.4953	0.3734	0.9128	0.0222	0.9366	0.0077	0.0990
	RMS	0.8925	0.0041	*0.8882*	0.0000	0.6288	0.1278	0.8334	0.0559	0.8756	0.0097	0.8859	**0.0047**	0.0337
	ROC	0.9258	**0.0158**	*0.9466*	0.0000	0.4275	0.0022	0.7746	0.0954	0.6845	0.1990	0.8418	0.0551	0.0612
	SAR	0.9255	**0.0069**	*0.9309*	0.0000	0.6736	-0.0037	0.8515	0.0538	0.8396	0.0695	0.8955	0.0165	0.0238
	MUC-F1	*0.6691*	**0.0000**	0.6242	0.0046	0.6405	0.0344	0.5340	0.1185	0.6500	0.0291	0.5181	0.1216	0.0514
	B3F1	*0.4625*	0.0000	0.4512	0.0000	0.4423	0.0425	0.0965	0.3357	0.4249	**0.0675**	0.3323	0.1430	0.0981
PSF identification	ACC	—	—	0.9854	0.0007	—	—	0.9873	0.0011	0.9862	0.0003	*0.9886*	**0.0000**	0.0005
	APR	—	—	0.6430	0.0316	—	—	0.5588	0.1948	0.4335	0.0381	*0.7697*	**0.0372**	0.0754
	BEP	—	—	0.5954	0.0165	—	—	0.6727	0.0302	0.4436	0.0718	*0.6961*	**0.0385**	0.0393
	F1	—	—	0.5643	0.0276	—	—	0.6837	0.0019	0.5770	0.0367	*0.6741*	**0.0062**	0.0181
	MXE	—	—	0.9342	0.0029	—	—	0.8089	0.1425	0.9265	0.0062	*0.9572*	**0.0093**	0.0402
	RMS	—	—	0.8838	0.0028	—	—	0.8896	0.0118	0.8839	0.0020	*0.9000*	**0.0068**	0.0058
	ROC	—	—	0.9576	0.0121	—	—	0.8566	0.1149	0.7181	0.1593	*0.9659*	**0.0188**	0.0763
	SAR	—	—	0.9329	0.0052	—	—	0.9021	0.0407	0.8541	0.0532	*0.9420*	**0.0085**	0.0269
	GACC	—	—	0.6607	-0.0004	—	—	*0.7962*	**0.0223**	0.6610	0.0506	0.7401	0.0209	0.0233
	F0.5	—	—	0.6829	0.0221	—	—	0.7150	0.0503	0.7132	0.0000	*0.7811*	**-0.0054**	0.0167
	F0.2	—	—	0.7701	0.0157	—	—	0.7331	0.0875	*0.8171*	**0.0110**	0.8542	0.0045	0.0297
PSF hierarchy	ACC	*0.8133*	**0.0000**	0.7554	0.0009	—	—	0.7940	0.0000	0.7446	0.0428	0.7761	0.0381	0.0164
	APR	0.8166	**0.0296**	0.7455	0.0013	—	—	*0.8035*	0.0000	0.5957	0.1996	0.6363	0.1520	0.0765
	BEP	*0.7385*	**-0.0066**	0.6597	-0.0030	—	—	0.7096	0.0000	0.6317	0.0567	0.6940	0.0432	0.0181
	F1	*0.7286*	**0.0033**	0.6810	0.0226	—	—	0.7000	0.0000	0.6774	0.0525	0.6933	0.0400	0.0237
	MXE	*0.6091*	**0.0166**	0.4940	0.0076	—	—	0.0379	0.4715	0.4022	0.1197	0.4681	0.1012	0.1433
	RMS	*0.6475*	**0.0054**	0.5910	0.0033	—	—	0.6057	0.0000	0.5556	0.0514	0.5836	0.0423	0.0205
	ROC	*0.8923*	**0.0096**	0.8510	0.0000	—	—	0.8519	0.0364	0.7514	0.1094	0.7968	0.0757	0.0462
	SAR	*0.7765*	**0.0073**	0.7251	0.0009	—	—	0.7430	0.0000	0.6770	0.0672	0.7116	0.0482	0.0247
	SENTACC	*0.7571*	**0.0045**	0.7307	-0.0011	—	—	0.7399	-0.0007	0.6801	0.0141	0.6889	0.0726	0.0179
	PERFSENT	*0.4948*	**0.0069**	0.4880	0.0000	—	—	0.4880	-0.0034	0.4055	0.0206	0.4158	0.1031	0.0254

Table 2: Performance gains from parameter tuning. The left column for each algorithm family is the algorithm's performance with default parameter settings. The adjacent 'Parm∆' column gives the performance gain from tuning parameters. For each metric, the best default and tuned performance across all algorithms are *italicized* and **bold-faced**, respectively.

5.1 Experiment 1: Parameter Tuning

Experiment 1 measures, for each of the 3 tasks, the performance of every model on both the tuning and test data for every metric of interest. *Based on tuning set performance,* the best default model, the best model overall, and the best model within each family are selected. The **best default model** is the highest-scoring model that emerges after comparing algorithms without doing any parameter tuning. The **best overall model** corresponds to "tuning" both the algorithm and parameters. The **best model in each family** corresponds to "tuning" the parameters for that algorithm. *Using the test set performances,* the best family models are compared to the corresponding default models to find the gains from parameter tuning.

Table 2 lists the gains achieved by parameter tuning. Each algorithm column compares the algorithm's best model to its default model. On the coreference task, for example, the best KNN model with respect to BEP shows a 20% improvement (or gain) over the default KNN model (for a final BEP score of .5300). The "Avg ∆" column shows the average gain from parameter tuning for each metric.

For each metric, the best default model is italicized while the best overall model is bold-faced. Referring again to the coreference BEP row, the best overall model is a SVM while the best default model is a bagged decision tree. Recall that these distinctions are based on *absolute performance* and not gain. Thus, the best tuned SVM outperforms all other models on this task and metric.[6]

Three conclusions can be drawn from Table 2. *First, no algorithm performs the best on all tasks or on all metrics.* For coreference, the best overall model is either a bagged tree, a rule learner, or a SVM, depending on the target metric. Similarly, for PSF identification the best model depends on the metric, ranging from a KNN to a SVM. Interestingly, bagged decision trees on the PSF hierarchy data outperform the other algorithms on all metrics and seem especially well-suited to the task.

Second, an algorithm's best-tuned model reliably yields non-trivial gains over the corresponding default model. This trend appears to hold regardless of algorithm, metric, and data set. In 114 of the 154

[6] Another interesting example is the best overall model for BEP on the PSF hierarchy task. The baseline (a bagged tree) outperforms the "best" model (a different bagged tree) on the test set even though the best model performed better on the tuning set—otherwise it would not have been selected as the best.

cases parameter tuning improves an algorithm's performance by more than 0.001 (0.1%). In the remaining 40 cases, parameter tuning only hurts 12 times, and never by more than 0.01.

Third, the best default algorithm is not necessarily the best algorithm after tuning parameters. The coreference task, in particular, illustrates the potential problem with using default parameter settings when judging which algorithm is most suited for a problem: 7 out of 10 times the best algorithm changes after parameter tuning.

These results corroborate those found elsewhere (Daelemans and Hoste, 2002; Hoste *et al.*, 2002; Hoste, 2005)—parameter settings greatly influence performance. Further, algorithmic performance differences can change when parameters are changed. Going beyond previous work, our results also underline the need to consider multiple algorithms for NLL. Ultimately, it is important for researchers to thoroughly explore options for **both** algorithm and parameter tuning and to report these in their results.

5.2 Experiment 2: Ensemble Selection

In experiment 2 an ensemble is built to optimize each target metric. The ensemble's performance is compared to that of the best overall model for the metric. Both the ensemble and the best model are selected according to tuning set performance.

Table 3 lists the gains from ensemble selection over the best parameter tuned model. For comparison, the best default and overall performances from Table 2 are reprinted. For example, the ensemble optimized for F1 on the coreference data outperforms the best bagged tree model by about 1% (and the best default model by almost 8%).

Disappointingly, ensemble selection does not consistently improve performance. Indeed, for the PSF hierarchy task ensemble selection reliably hurts performance a small amount. For the other two tasks ensemble selection reliably improves all metrics except GACC (a small loss). In other experiments, however, we optimized F-measure with $\beta = 1.5$ for the PSF identification task. Ensemble selection hurt F1.5 by almost 2%, leading us to question the technique's reliability for our second data set. Interestingly, the aggregate metrics—metrics that measure performance by combining multiple predictions—all benefit from ensemble selection, even for the hierarchy task, *albeit* for small amounts. For our tasks these comprise a subset of the task-specific performance measures: B3F1, MUC-F1, SENTACC, and PERFSENT.

	Metric	Best Default	Best TunedΔ	Ens. Sel.Δ
coreference	ACC	0.9861	0.0002	**0.0001**
	APR	0.5373	0.0000	**0.0736**
	BEP	0.6010	0.0000	**0.0124**
	F1	0.5231	0.0664	**0.0115**
	MXE	0.9373	0.0142	**0.0035**
	RMS	0.8882	0.0023	**0.0049**
	ROC	0.9466	-0.0051	**0.0120**
	SAR	0.9309	0.0015	**0.0032**
	MUC-F1	0.6691	0.0000	**0.0073**
	B3F1	0.4625	0.0299	**0.0077**
PSF identification	ACC	0.9886	0.0000	**0.0003**
	APR	0.7697	0.0372	**0.0109**
	BEP	0.6961	0.0385	**0.0136**
	F1	0.6741	0.0062	**0.0222**
	MXE	0.9572	0.0093	**0.0029**
	RMS	0.9000	0.0068	**0.0025**
	ROC	0.9659	0.0188	**0.0043**
	SAR	0.9420	0.0085	**0.0021**
	GACC	0.7962	**0.0223**	-0.0012
	F0.5	0.7811	-0.0054	**0.0063**
	F0.2	0.8171	0.0110	**0.0803**
PSF hierarchy	ACC	**0.8133**	**0.0000**	-0.0028
	APR	0.8035	**0.0427**	-0.0064
	BEP	**0.7385**	-0.0066	0.0056
	F1	0.7286	**0.0033**	-0.0016
	MXE	0.6091	**0.0166**	-0.0012
	RMS	0.6475	**0.0054**	-0.0019
	ROC	0.8923	**0.0096**	-0.0036
	SAR	0.7765	**0.0073**	-0.0015
	SENTACC	0.7571	0.0045	**0.0024**
	PERFSENT	0.4948	0.0069	**0.0172**

Table 3: Impact from tuning and ensemble selection. *Best default* shows the performance of the best classifier with no parameter tuning (*i.e.* algorithm tuning only). *Best tuned*Δ gives the performance gain from parameter and algorithm tuning. *Ens. Sel.*Δ is the performance gain from ensemble selection over the best tuned model. The best performance for each metric is marked in bold.

While we are not surprised that the positive gains are small,[7] we are surprised at how often ensemble selection hurts performance. As a result, we investigated some of the metrics where ensemble selection hurts performance and found that ensemble selection overfits the tuning data. At this time we are not sure why this overfitting happens for these tasks and not for the ones used by Caruana *et al*. Preliminary investigations suggest that having a smaller model library is a contributing factor (Caruana *et al.* use libraries containing $\sim$ 2000 models). This is consistent with the fact that the task with the largest model library, coreference, benefits the most from ensemble selection. Perhaps the reason that ensemble selection consistently improves performance for

[7] Caruana *et al.* (2004) find the benefit from ensemble selection is only half as large as the benefit from carefully optimizing and selecting the best models in the first place.

the aggregate metrics is that these metrics are harder to overfit.

Based on our results, ensemble selection seems too unreliable for general use in NLL—at least until the model library requirements are better understood. However, ensemble selection is perhaps trustworthy enough to optimize metrics that are difficult to overfit and could not be easily optimized otherwise — in our case, the task-specific aggregate performance measures.

6 Related Work

Hoste *et al.* (2002) and Hoste (2005) study the impact of tuning parameters for k-NN and a rule-learning algorithm on word sense disambiguation and coreference resolution, respectively, and find that parameter settings greatly change results. Similar work by Daelemans and Hoste (2002) shows the fallacy of comparing algorithm performance without first tuning parameters. They find that the best algorithm for a task frequently changes after optimizing parameters. In contrast to our work, these earlier experiments investigate at most two algorithms and only measure performance with one metric per task.

7 Conclusion

We evaluate an ensemble selection framework that enables optimizing classifier performance to arbitrary performance metrics without re-training. An important side benefit of the framework is the fully automatic production of base-level models, removing the need for human expertise in choosing algorithms and parameter settings.

Our experiments show that ensemble selection, compared to simple algorithm and parameter tuning, reliably improves performance for six of the seven task-specific metrics and all four "aggregate" metrics, but only benefits *all* of the metrics for one of our three data sets. We also find that exploring multiple algorithms with a variety of settings is important for getting the best performance. Tuning parameter settings results in 0.05% to 14% average improvements, with most improvements falling between 2% and 10%. To this end, the ensemble selection *framework* can be used as an environment for automatically choosing the best algorithm and parameter settings for a given NLP classification task. More work is needed to understand when ensemble selection can be safely used for NLL.

Acknowledgements We thank Alex Niculescu-Mizil for answering questions about ensemble selection and Eric Breck for sharing his expertise with both PSF data sets. We also thank the anonymous reviewers and the Cornell NLP group for helpful suggestions on paper drafts. This work was supported by the Advanced Research and Development Activity (ARDA), by NSF Grant IIS-0208028, and by NSF Grant IIS-0412930.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Linguistic Coreference Workshop at LREC 1998*, pages 563–566, May.

Eric Breck and Claire Cardie. 2004. Playing the telephone game: Determining the hierarchical structure of perspective and speech expressions. In *COLING 2004*, pages 120–126.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Wray Buntine. 1993. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, pages 182–201. Chapman & Hall, London.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *ICML*.

William W. Cohen. 1995. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *ICML*, pages 115–123, Tahoe City, CA, July 9–12. Morgan Kaufmann.

Walter Daelemans and Véronique Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *LREC 2002*, pages 755–760.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2000. TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, Tilburg University. Available from `http://ilk.kub.nl/˜ilk/papers/ ilk0001.ps.gz`.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *ICML*, pages 148–156.

Véronique Hoste, Iris Hendrickx, Walter Daelemans, and Antal van den Bosch. 2002. Parameter optimization for machine learning of word sense disambiguation. *Natural Language Engineering*, 8(4):311–325.

Véronique Hoste. 2005. *Optimization Issues in Machine Learning of Coreference Resolution*. Ph.D. thesis, University of Antwerp.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, USA.

Miroslav Kubat and Stan Matwin. 1997. Addressing the curse of imbalanced training sets: One-sided selection. In *ICML*, pages 179–186, San Francisco, CA. Morgan Kaufmann.

Andrew Kachites McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

MPQA Corpus. 2002. NRRC MPQA corpus. Available from `http://nrrc.mitre.org/NRRC/Docs_Data/MPQA_04/approval_time.htm`.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *ACL*, pages 104–111.

Vincent Ng. 2004. *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In Alexander J. Smola, Peter J. Bartlett, Bernhard Schoelköpf, and Dale Schuurmans, editors, *Advances in Large-Margin Classifiers*, pages 61–74. MIT Press.

Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of the 6th Message Understanding Conference*, pages 45–52. Morgan Kaufmann.

Janyce Wiebe, Eric Breck, Chris Buckley, Claire Cardie, Paul Davis, Bruce Fraser, Diane Litman, David Pierce, Ellen Riloff, Theresa Wilson, David Day, and Mark Maybury. 2003. Recognizing and organizing opinions expressed in the world press. In *Papers from the AAAI Spring Symposium on New Directions in Question Answering (AAAI tech report SS-03-07)*. March 24-26, 2003. Stanford University, Palo Alto, California.

Ian H. Witten and Eibe Frank. 2000. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, CA.