Why Nitpicking Works: Evidence for Occam's Razor in Error Correctors

Dekai WU[†1] dekai@cs.ust.hk
Grace NGAI[‡2] csgngai@polyu.edu.hk
Marine CARPUAT[†] marine@cs.ust.hk

[†] Human Language Technology Center
HKUST
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong

[‡] Hong Kong Polytechnic University
Department of Computing
Kowloon
Hong Kong

Abstract

Empirical experience and observations have shown us when powerful and highly tunable classifiers such as maximum entropy classifiers, boosting and SVMs are applied to language processing tasks, it is possible to achieve high accuracies, but eventually their performances all tend to plateau out at around the same point. To further improve performance, various error correction mechanisms have been developed, but in practice, most of them cannot be relied on to predictably improve performance on unseen data; indeed, depending upon the test set, they are as likely to degrade accuracy as to improve it. This problem is especially severe if the base classifier has already been finely tuned.

In recent work, we introduced N-fold Templated Piped Correction, or NTPC ("nitpick"), an intriguing error corrector that is designed to work in these extreme operating conditions. Despite its simplicity, it consistently and robustly improves the accuracy of existing highly accurate base models. This paper investigates some of the more surprising claims made by NTPC, and presents experiments supporting an Occam's Razor argument that more complex models are damaging or unnecessary in practice.

1 Introduction

The investigation we describe here arose from a very commonly discussed experience, apparently triggered by the recent popularity of shared task evaluations that have opened opportunities for researchers to informally compare their experiences "with a common denominator", so to speak.

Among the perennial observations which are made during the analysis of the results is that (1) methods designed to "fine-tune" the high-accuracy base classifiers behave unpredictably, their success or failure often appearing far more sensitive to where the test set was drawn from, rather than on any true quality of the "fine-tuning", and consequently, (2) the resulting system rankings are often unpredictable, especially as they are typically conducted only on a single new test set, often drawn from a single arbitrary new source of a significantly different nature than the training sets. One could argue that such evaluations do not constitute a fair test, but in fact, this is where computational linguistics modeling diverges from machine learning theory, since for any serious NLP application, such evaluations constitute a much more accurate representation of the real world.

We believe one primary reason for this common experience is that the models involved are typically already operating well beyond the limits of accuracy of the models' assumptions about the nature of distributions from which testing samples will be drawn. For this reason, even "sophisticated" discriminative training criteria, such as maximum entropy, minimum error rate, and minimum Bayes risk, are susceptible to these stability problems. There has been much theoretical work done on error correction, but in practice, any error correction usually lowers the performance of the combined system on unseen data, rather than improving it. Unfortunately, most existing theory simply does not apply.

This is especially true if the base model has been highly tuned. For the majority of tasks, the performance of the trained models, after much fine tuning, tend to plateau out at around the same point, regardless of the theoretical basis of the underlying model. This holds true with most highly accurate classifiers, including maximum entropy classifiers, SVMs, and boosting models. In addition, even though data analysis gives us some general idea as to what kinds of feature conjunctions might help, the classifiers are not able to incorporate those into their model (usually because the computational cost would be infeasible), and any further post-processing tends to degrade accuracy on unseen data. The common practice of further improving accuracy at this point is to resort to *ad hoc* classifier combination methods, which are usually not theoretically well justified and, again, unpredictably improve or degrade performance—thus consuming vast amounts of experimental resources with relatively low expected payoff, much like a lottery.

There are a variety of reasons for this, ranging from the aforementioned validity of the assumptions about the distribution between the training and test corpora, to the absence of a well justified stopping point for error correction. The latter problem is much more serious than it seems at first blush, since without a well-justified stopping criterion, the performance of the combined model will be much more dependent upon the distribution of the test set, than on any feature engineering. Empirical evidence for this argument can be seen from the result of the CoNLL shared tasks (Tjong Kim Sang, 2002)(Tjong Kim Sang and Meulder, 2003), where the ranking of the

[1]The author would like to thank the Hong Kong Research Grants Council (RGC) for supporting this research in part through research grants RGC6083/99E, RGC6256/00E, and DAG03/04.EG09.

[2]The author would like to thank the Hong Kong Polytechnic University for supporting this research in part through research grants A-PE37 and 4-Z03S.

participating systems changes with the test corpora.

Inspired by the repeated observations of this phenomenon by many participants, we decided to stop "sweeping the issue under the rug", and undertook to confront it head-on. Accordingly, we challenged ourselves to design an error corrector satisfying the following criteria, which few if any existing models actually meet: (1) it would leverage off existing base models, while targeting their errors; (2) it would *consistently* improve accuracy, even on top of base models that already deliver high accuracy; (3) it would be robust and conservative, so as to almost *never* accidentally degrade accuracy; (4) it would be broadly applicable to any classification or recognition task, especially high-dimensional ones such as named-entity recognition and word-sense disambiguation; and (5) it would be template-driven and easily customizable, which would enable it to target error patterns beyond the base models' representation and computational complexity limitations.

Our goal in this undertaking was to invent as little as possible. We expected to make use of relatively sophisticated error-minimization techniques. Thus the results were surprising: the *simplest* models kept outperforming the "sophisticated" models. This paper attempts to investigate some of the key reasons why.

To avoid reinventing the wheel, we originally considered adapting an existing error-driven method, transformation-based learning (TBL) for this purpose. TBL seems well suited to the problem as it is inherently an error corrector and, on its own, has been shown to achieve high accuracies on a variety of problems (see Section 4). Our original goal was to adapt TBL for error correction of high-performing models (Wu *et al.*, 2004a), with two main principles: (1) since it is not clear that the usual assumptions made about the distribution of the training/test data are valid in such extreme operating ranges, empirical observations would take precedence over theoretical models, which implies that (2) any model would have to be empirically justified by testing on a diverse range of data. Experimental observations, however, increasingly drove us toward different goals.

Our resulting error corrector, NTPC, was instead constructed on the principle of making as few assumptions as possible in order to robustly generalize over diverse situations and problems. One observation made in the course of experimentation, after many attempts at fine-tuning model parameters, was that many of the complex theoretical models for error correction often do not perform consistently. This is perhaps not too surprising upon further reflection, since the principle of Occam's Razor does prefer simpler hypotheses over more complex ones.

NTPC was introduced in (Wu *et al.*, 2004b), where the controversial issues it raised generated a number of interesting questions, many of which were were directed at NTPC's seeming simplicity, which seems in opposition to the theory behind many other error correcting models. In this paper, we investigate the most commonly-asked questions. We illuminate these questions by contrasting NTPC against the more powerful TBL, presenting experiments that show that NTPC's simple model is indeed key to its robustness and reliability.

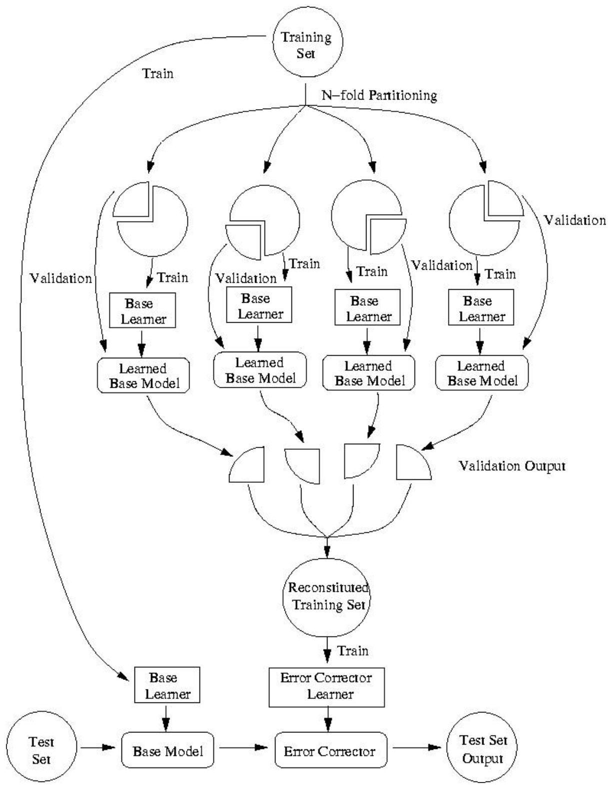

Figure 1: Piped architecture with n-fold partitioning.

The rest of the paper is laid out as follows: Section 2 presents an introduction to NTPC, including an overview of its architecture. Section 3 addresses key questions related to NTPC's architecture and presents empirical results justifying its simplicity.

2 N-fold Templated Piped Correction

N-fold Templated Piped Correction or **NTPC**, is a model that is designed to robustly improve the accuracy of existing base models in a diverse range of operating conditions. As was described above, the most challenging situations for any error corrector is when the base model has been finely tuned and the performance has reached a plateau. Most of the time, any further feature engineering or error correction after that point will end up hurting performance rather than improving it.

2.1 The architecture of NTPC is surprisingly simple

One of the most surprising things about NTPC lies in the fact that despite its simplicity, it outperforms mathematically much more "sophisticated" methods at error correcting. Architecturally, it relies on a simple rule-learning mechanism and cross-partitioning of the training data to learn very conservative, cautious rules that make only a few corrections at a time.

Figure 1 illustrates the NTPC architecture. Prior to learning, NTPC is given (1) a set of rule templates which describe the types of rules that it is allowed to hypothesize, (2) a single base learning model, and (3) an annotated training set.

The NTPC architecture is essentially a sequentially chained **piped** ensemble that incorporates cross-validation style n-fold partition sets generated from the base model. The training set is partitioned n times in order to train n base models. Subsequently the n held-out validation sets are classified by the respective trained base models, with the results combined into a "reconstituted" training set. The reconstituted training set is used by *Error_Corrector_Learner*, which learns a set of rules. Rule hypotheses are generated according to the given set of allowable templates:

$$\mathcal{R} = \{r | r \in \mathcal{H} \wedge \tau(r) > \tau_{\min} \wedge \epsilon(r) = 0\} \quad (1)$$

$$\tau(r) = \sum_{j=1}^{X} \sum_{r(x_j,\hat{y}_j)\neq\emptyset} \delta(r(x_j,\hat{y}_j), y_j) \quad (2)$$

$$\epsilon(r) = \sum_{j=1}^{X} \sum_{r(x_j,\hat{y}_j)\neq\emptyset} 1 - \delta(r(x_j,\hat{y}_j), y_j) \quad (3)$$

where $\mathcal{X}$ is a sequence of X training examples x_i, $\mathcal{Y}$ is a sequence of reference labels y_i for each example respectively, $\hat{\mathcal{Y}}$ is a sequence of labels $\hat{y}_i$ as predicted by the base model for each example respectively, $\mathcal{H}$ is the hypothesis space of valid rules implied by the templates, and $\tau_{\min}$ is a confidence threshold. Setting $\tau_{\min}$ to a relatively high value (say 15) implements the requirement of high reliability. $\mathcal{R}$ is subsequently sorted by the τ_i value of each rule r_i into an ordered list of rules $\mathcal{R}^* = (r_0^*, \ldots, r_{i-1}^*)$.

During the evaluation phase, depicted in the lower portion of Figure 1, the test set is first labeled by the base model. The error corrector's rules r_i^* are then applied in the order of R^* to the evaluation set. The final classification of a sample is then the classification attained when all the rules have been applied.

2.2 NTPC consistently and robustly improves accuracy of highly-accurate base models

In previous work (Wu *et al.*, 2004b), we presented experiments on named-entity identification and classification across four diverse languages, using Adaboost.MH as the base learner, which showed that NTPC was capable of robustly and consistently improving upon the accuracy of the already-highly-accurate boosting model; correcting the errors committed by the base model but not introducing any of its own.

Table 1 compares results obtained with the base Adaboost.MH model (Schapire and Singer, 2000) and the NTPC-enhanced model for a total of eight different named-entity recognition (NER) models. These experiments were performed on the CoNLL-2002 and CoNLL-2003 shared task data sets. It can be seen that the Adaboost.MH base models clearly already achieve high accuracy, setting the bar very high for NTPC to improve upon. However, it can also be seen that NTPC yields further F-Measure gains on *every* combination of task and language, including English NE bracketing (Model M2) for which the base F-Measure is the highest.

An examination of the rules (shown in the Appendix) can give an idea as to why NTPC manages to identify and correct errors which were overlooked by the highly tuned base model. NTPC's advantage comes from two aspects: (1) its ability to handle complex conjunctions of features, which often reflect structured, linguistically motivated expectations, in the form of rule templates; and (2) its ability to "look forward" at classifications from the right context, even when processing the sentence in a left-to-right direction. The base classifier is unable to incorporate these two aspects, because (1) including complex conjunctions of features would raise the computational cost of searching the feature space to a point where it would be infeasible, and (2) most classifiers process a sentence from left-to-right, deciding on the class label for each word before moving on to the next one. Rules that exploit these advantages are easily picked out in the table; many of the rules (especially those in the top 5 for both English and Spanish) consist of complex conjunctions of features; and rules that consider the right context classifications can be identified by the string "ne_<*num*>", where <*num*> is a positive integer (indicating how many words to the right).

3 Experiments

The most commonly-raised issues about NTPC relate to the differences between NTPC and TBL (though the conceptual issues are much the same as for other error-minimization criteria, such as minimum error rate or minimum Bayes risk). This is expected, since it was one of our goals to reinvent as little as possible. As a result, NTPC does bear a superficial resemblance to TBL, both of them being error-driven learning methods that seek to incrementally correct errors in a corpus by learning rules that are determined by a set of templates. One of the most frequently asked questions is whether the *Error_Corrector_Learner* portion of NTPC could be replaced by a transformation-based learner. This section will investigate the differences between NTPC and TBL, and show the necessity of the changes that were incorporated into NTPC.

The experiments run in this section were performed on the data sets used in the CoNLL-2002 and CoNLL-2003 Named Entity Recognition shared tasks. The high-performing base model is based on AdaBoost.MH (Schapire and Singer, 2000), the multi-class generalization of the original boosting algorithm, which implements boosting on top of decision stump classifiers (decision trees of depth one).

3.1 Any Error is Bad

The first main difference between NTPC and TBL, and also what seems to be an extreme design decision on the part of NTPC, is the objective scoring function. To be maximally certain of not introducing any new errors with its rules, the first requirement that NTPC's objective function places onto any candidate rules is that they must not introduce any new errors ($\epsilon(r) = 0$). This is called the *zero error tolerance* principle.

To those who are used to learners such as transformation-based learning and decision lists, which allow for some degree of error tolerance, this design principle seems overly harsh and inflexible. Indeed, for al-

Table 1: NTPC consistently yields improvements on all eight different high-accuracy NER base models, across every combination of task and language.

Model	Task	Language	Model	Precision	Recall	F-Measure$_1$
M1	Bracketing	Dutch	Base	87.27	91.48	89.33
			Base w/ NTPC	**87.44**	**92.04**	**89.68**
M2	Bracketing	English	Base	95.01	93.98	94.49
			Base w/ NTPC	**95.23**	**94.05**	**94.64**
M3	Bracketing	German	Base	**83.44**	65.86	73.62
			Base w/ NTPC	83.43	**65.91**	**73.64**
M4	Bracketing	Spanish	Base	89.46	87.57	88.50
			Base w/ NTPC	**89.77**	**88.07**	**88.91**
M5	Classification + Bracketing	Dutch	Base	70.26	73.64	71.91
			Base w/ NTPC	**70.27**	**73.97**	**72.07**
M6	Classification + Bracketing	English	Base	88.64	87.68	88.16
			Base w/ NTPC	**88.93**	**87.83**	**88.37**
M7	Classification + Bracketing	German	Base	**75.20**	59.35	66.34
			Base w/ NTPC	75.19	**59.41**	**66.37**
M8	Classification + Bracketing	Spanish	Base	74.11	72.54	73.32
			Base w/ NTPC	**74.43**	**73.02**	**73.72**

most all models, there is an implicit assumption that the scoring function will be based on the difference between the positive and negative applications, rather than on an absolute number of corrections or mistakes.

Results for eight experiments are shown in Figures 2 and 3. Each experiment compares NTPC against other variants that allow relaxed $\epsilon(r) \leq \epsilon_{\max}$ conditions for various $\epsilon_{\max} \in \{1, 2, 3, 4, \infty\}$. The worst curve in each case is for $\epsilon_{\max} = \infty$ — in other words, the system that only considers net performance improvement, as TBL and many other rule-based models do. The results confirm empirically that the $\epsilon(r) = 0$ condition (1) gives the most consistent results, and (2) generally yields accuracies among the highest, regardless of how long training is allowed to continue. In other words, the presence of *any* negative application during the training phase will cause the error corrector to behave unpredictably, and the more complex model of greater error tolerance is unnecessary in practice.

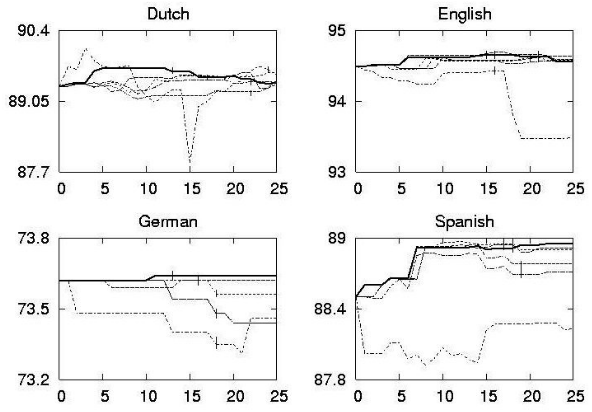

Figure 2: NTPC's zero tolerance condition yields less fluctuation and generally higher accuracy than the relaxed tolerance variations, in bracketing experiments. (bold = NTPC, dashed = relaxed tolerance)

3.2 Rule Interaction is Unreliable

Another key difference between NTPC and TBL is the process of rule interaction. Since TBL allows a rule to use the current classification of a sample and its neighbours as features, and a rule updates the current state of the corpus when it applies to a sample, the application of one rule could end up changing the applicability (or not) of another rule. From the point of view of a sample, its classification could depend on the classification of "nearby" samples. Typically, these "nearby" samples are those found in the immediately preceding or succeeding words of the same sentence. This rule interaction is permitted in both training and testing.

NTPC, however, does not allow for this kind of rule interaction. Rule applications only update the output classification of a sample, and do not update the current state of the corpus. In other words, the feature values for a sample are initialized once, at the beginning of the program, and not changed again thereafter. The rationale for making this decision is the hypothesis that rule interaction is in nature unreliable, since the high-accuracy base model provides sparse opportunities for rule application and thus much sparser opportunities for rule interaction, making any rule that relies on rule interaction suspect. As a matter of fact, by considering only rules that make no mistake during the learning phase, NTPC's zero error tolerance already eliminates any correction of labels that results from rule interaction—since a label correction on a sample that results from the application of more than one rule necessarily implies that at least one of the rules made a mistake.

Since TBL is a widely used error-correcting method,

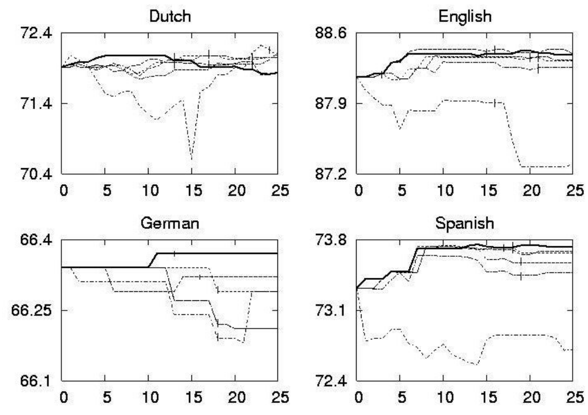

Figure 3: NTPC's zero tolerance condition yields less fluctuation and generally higher accuracy than the relaxed tolerance variations, in bracketing + classification experiments. (bold = NTPC, dashed = relaxed tolerance)

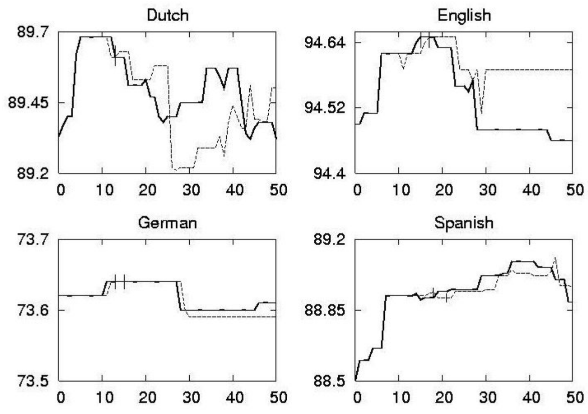

Figure 4: Unpredictable fluctuations on the bracketing task show that allowing TBL-style rule interaction does not yield reliable improvement over NTPC. (bold = NTPC, dashed = rule interaction)

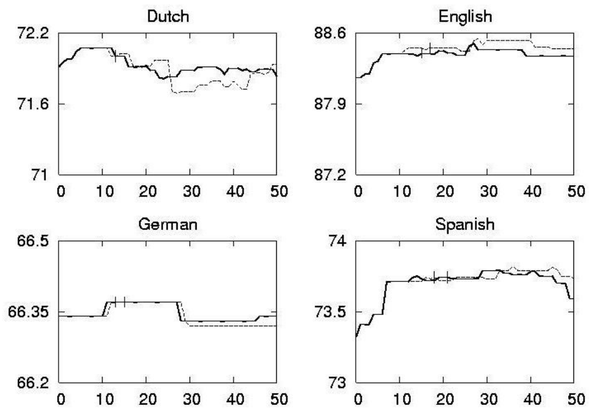

Figure 5: Unpredictable fluctuations on the bracketing + classification task show that allowing TBL-style rule interaction does not yield reliable improvement over NTPC. (bold = NTPC, dashed = rule interaction)

it is natural to speculate that NTPC's omission of rule interaction is a weakness. In order to test this question, we implemented an iterative variation of NTPC that allows rule interaction, where each iteration targets the residual error from previous iterations as follows:

```
1. $i \leftarrow 0, \mathcal{X}_0 \leftarrow \mathcal{X}$
2. $r_i^* \leftarrow$ null, $s_i^* \leftarrow 0$
3. **foreach** $r \in \mathcal{H}$ such that $\epsilon_i(r) = 0$
    • **if** $\tau_i(r) > \tau_i^*$ **then** $r_i^* \leftarrow r, \tau_i^* \leftarrow \tau_i(r)$
4. **if** $\tau_i^* < \tau_{\min}$ **then return**
5. $\mathcal{X}_{i+1} \leftarrow$ result of applying $r_i^*$ to $\mathcal{X}_i$
6. $i \leftarrow i + 1$
7. **goto** Step 3
```

where

$$\tau_i(r) = \sum_{j=1}^{X} \sum_{r(x_j^i, \hat{y}_j) \neq \emptyset} \delta(r(x_j^i, \hat{y}_j), y_j)$$

$$\epsilon_i(r) = \sum_{j=1}^{X} \sum_{r(x_j^i, \hat{y}_j) \neq \emptyset} 1 - \delta(r(x_j^i, \hat{y}_j), y_j)$$

Here, incremental rule interaction is a natural consequence of arranging the structure of the algorithm to observe the right context features coming from the base model, as with transformation-based learning. In Step 5 of the algorithm, the current state of the corpus is updated with the latest rule on each iteration. That is, in each given iteration of the outer loop, the learner considers the corrected training data obtained by applying rules learned in the previous iterations, so the learner has access to the labels that result from applying the previous rules. Since these rules may apply anywhere in the corpus, the learner is not restricted to using only labels from the left context.

The time complexity of this variation is an order of magnitude more expensive than NTPC, due to the need to allow rule interaction using nested loops. The ordered list of output rules $r_0^*, \ldots, r_{i-1}^*$is learned in a greedy fashion, to progressively improve upon the performance of the learning algorithm on the training set.

Results for eight experiments on this variation, shown in Figures 4 and 5, demonstrate that this expensive extra capability is rarely useful in practice and does not reliably guarantee that accuracy will not be degraded. This is yet another illustration of the principle that, in high-accuracy error correction problems, at least, more simple modes of operation should be preferred over more complex arrangements.

3.3 NTPC vs. N-fold TBL

Another question on NTPC that is frequently raised is whether or not ordinary TBL, which is after all, intrinsically an error-correcting model, can be used in place of NTPC to perform better error correction. Figure 6 shows the results of four sets of experiments evaluating this approach on top of boosting. As might be expected from extrapolation from the foregoing experiments that investigated their individual differences, NTPC outperforms the more complex TBL in all cases, regardless of how long training is allowed to continue.

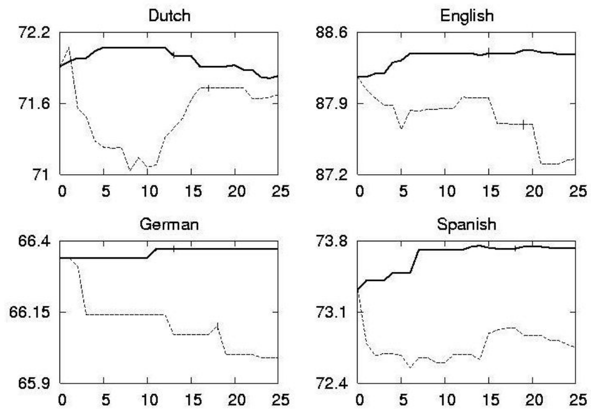

Figure 6: NTPC consistently outperforms error correction using TBL even when n-fold partitioning is used. (bold = NTPC, dashed = TBL with n-fold partitioning)

Table 2: The more complex partition-based voted error corrector degrades performance, while NTPC helps (bracketing + classification, English).

Model	**Precision**	**Recall**	**F-Measure$_1$**
Base	95.01	93.98	94.49
Partition-Based Voting	95.07	93.79	94.43
Base w/ NTPC	**95.14**	**94.05**	**94.59**

3.4 NTPC vs. Partition-Based Voting

Another valid question would be to ask if the way that NTPC combines the results of the n-fold partitioning is oversimplistic and could be improved upon. As was previously stated, the training corpus for the error corrector in NTPC is the "reconstituted training set" generated by combining the held-out validation sets after they have labeled with initial classifications by their respective trained base models. To investigate if NTPC could benefit from a more complex model, we employed voting, a commonly-used technique in machine learning and natural language processing. As before, the training set was partitioned and multiple base learners were trained and evaluated on the multiple training and validation sets, respectively. However, instead of recombining the validation sets into a reconstituted training set, multiple error corrector models were trained on the n partition sets. During the evaluation phase, all n error correctors were evaluated on the evaluation set after it had been labeled by the base model, and they voted on the final output.

Table 2 shows the results of using such an approach for the bracketing + classification task on English. The empirical results clearly show that the more complex and time-consuming voting model not only does not outperfom NTPC, but in fact again *degrades* the performance from the base boosting-only model.

3.5 Experiment Summary

In our experiments, we set out to investigate whether NTPC's operating parameters were overly simple, and whether more complex arrangements were necessary or desirable. However, empirical evidence points to the fact that, in this problem of error correction in high accuracy ranges, at least, simple mechanisms will suffice to produce good results—in fact, the more complex operations end up degrading rather than improving accuracy.

A valid question is to ask why methods such as decision list learning (Rivest, 1987) as well as transformation-based learning benefit from these more complex mechanisms. Though structurally similar to NTPC, these models operate in a very different environment, where many initially poorly labeled examples are available to drive rule learning with. Hence, it is possibly advantageous to trade off some corrections with some mistakes, provided that there is an overall positive change in accuracy. However, in an error-correcting situation, most of the samples are already correctly labeled, errors are few and far in between and the sparse data problem is exacerbated. In addition, the idea of error correction implies that we should, at the very least, not do any worse than the original algorithm, and hence it makes sense to err on the side of caution and minimize any errors created, rather than hoping that a later rule application will undo mistakes made by an earlier one.

Finally, note that the same point applies to many other models where training criteria like minimum error rate are used, since such criteria are functions of the tradeoff between correctly and incorrectly labeled examples, without zero error tolerance to compensate for the sparse data problem.

4 Previous Work

4.1 Boosting and NER

Boosting (Freund and Schapire, 1997) has been successfully applied to several NLP problems. In these NLP systems boosting is typically used as the ultimate stage in a learned system. For example, Shapire and Singer (2000) applied it to Text Categorization while Escudero *et al.*(2000) used it to obtain good results on Word Sense Disambiguation. More closely relevant to the experiments described here in, two of the best-performing three teams in the CoNLL-2002 Named Entity Recognition shared task evaluation used boosting as their base system (Carreras *et al.*, 2002)(Wu *et al.*, 2002).

However, precedents for improving performance *after* boosting are few. At the CoNLL-2002 shared task session, Tjong Kim Sang (unpublished) described an experiment using voting to combine the NER outputs from the shared task participants which, predictably, produced better results than the individual systems. A couple of the individual systems were boosting models, so in some sense this could be regarded as an example.

Tsukamoto *et al.*(2002) used piped AdaBoost.MH models for NER. Their experimental results were somewhat disappointing, but this could perhaps be attributable to various reasons including the feature engineering or not using cross-validation sampling in the stacking.

Appendix

The following examples show the top 10 rules learned for English and Spanish on the bracketing + classification task. (Models M6 and M8)

English
ne_-2=ZZZ ne_-1=ZZZ word:[1,3]=21 nonnevocab_0=inNonNeVocab nevocab_0=inNeVocab captype_0=firstword-firstupper => ne=I-ORG
ne_1=O ne_2=O word_-1=ZZZ nonnevocab_0=inNonNeVocab nevocab_0=not-inNeVocab captype_0=firstword-firstupper => ne=O
captype_0=notfirstword-firstupper captype_-1=firstword-firstupper captype_1=number nonnevocab_0=inNonNeVocab nevocab_0=inNeVocab ne_0=I-LOC => ne=I-ORG
ne_-1=ZZZ ne_0=I-ORG word_1=, nonnevocab_0=not-inNonNeVocab nevocab_0=not-inNeVocab captype_0=allupper => ne=I-LOC
ne_0=I-PER word:[1,3]=0 nonnevocab_0=not-inNonNeVocab nevocab_0=not-inNeVocab captype_0=notfirstword-firstupper => ne=I-ORG
ne_0=I-ORG ne_1=O ne_2=O nonnevocab_0=inNonNeVocab nevocab_0=not-inNeVocab captype_0=alllower => ne=O
ne_0=I-PER ne_1=I-ORG => ne=I-ORG
ne_-1=ZZZ ne_0=I-PER word:[-3,-1]=ZZZ word:[1,3]=1 => ne=I-ORG
ne_0=I-ORG word:[-3,-1]=spd => ne=B-ORG
ne_-1=I-ORG ne_0=I-PER word:[1,3]=1 => ne=I-ORG
Spanish
wcaptype_0=alllower ne_-1=I-ORG ne_0=I-ORG ne_1=O => ne=O
captypeLex_-1=inLex captypeGaz_-1=not-inGaz wcaptype_-1=alllower ne_-1=O ne_0=O captypeLex_0=not-inLex captypeGaz_0=not-inGaz wcaptype_0=noneed-firstupper => ne=I-ORG
wcaptype_0=noneed-firstupper wcaptype_-1=noneed-firstupper wcaptype_1=alllower captypeLex_0=not-inLex captypeGaz_0=not-inGaz ne_0=O => ne=I-ORG
ne_0=O word_0=efe => ne=I-ORG
ne_-1=O ne_0=O word_1=Num word_2=. captypeLex_0=not-inLex captypeGaz_0=not-inGaz wcaptype_0=allupper => ne=I-MISC
pos_-1=ART pos_0=NCF wcaptype_0=noneed-firstupper ne_-1=O ne_0=O => ne=I-ORG
wcaptype_0=alllower ne_0=I-PER ne_1=O ne_2=O => ne=O
ne_0=O ne_1=I-MISC word_2=Num captypeLex_0=not-inLex captypeGaz_0=not-inGaz wcaptype_0=allupper => ne=I-MISC
ne_0=I-LOC word:[-3,-1]=universidad => ne=I-ORG
ne_1=O ne_2=O word_0=de captypeLex_0=not-inLex captypeGaz_0=inGaz wcaptype_0=alllower => ne=O

The AdaBoost.MH base model's high accuracy sets a high bar for error correction. Aside from brute-force *en masse* voting of the sort at CoNLL-2002 described above, we do not know of any existing post-boosting models that improve rather than degrade accuracy. We aim to further improve performance, and propose using a piped error corrector.

4.2 Transformation-based Learning

Transformation-based learning (Brill, 1995), or TBL, is one of the most successful rule-based machine learning algorithms. The central idea of TBL is to learn an ordered list of rules, each of which evaluates on the results of those preceding it. An initial assignment is made based on simple statistics, and then rules are greedily learned to correct the mistakes, until no net improvement can be made.

Transformation-based learning has been used to tackle a wide range of NLP problems, ranging from part-of-speech tagging (Brill, 1995) to parsing (Brill, 1996) to segmentation and message understanding (Day *et al.*, 1997). In general, it achieves state-of-the-art performances and is fairly resistant to overtraining.

5 Conclusion

We have investigated frequently raised questions about N-fold Templated Piped Correction (NTPC), a general-purpose, conservative error correcting model, which has been shown to reliably deliver small but consistent gains on the accuracy of even high-performing base models on high-dimensional NLP tasks, with little risk of accidental degradation. Experimental evidence shows that when error-correcting high-accuracy base models, simple models and hypotheses are more beneficial than complex ones, while the more complex and powerful models are surprisingly unreliable or damaging in practice.

References

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

Eric Brill. *Recent Advances in Parsing Technology*, chapter Learning to Parse with Transformations. Kluwer, 1996.

Xavier Carreras, Lluís Màrques, and Lluís Padró. Named entity extraction using adaboost. In Dan Roth and Antal van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 167–170. Taipei, Taiwan, 2002.

David Day, John Aberdeen, Lynette Hirshman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C., March 1997. Association of Computational Linguistics.

Gerard Escudero, Lluis Marquez, and German Rigau. Boosting applied to word sense disambiguation. In *European Conference on Machine Learning*, pages 129–141, 2000.

Yoram Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences, 55(1)*, pages 119–139, 1997.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

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

Erik Tjong Kim Sang and Fien Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*. Edmonton, Canada, 2003.

Erik Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In Dan Roth and Antal van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan, 2002.

Koji Tsukamoto, Yutaka Mitsuishi, and Manabu Sassano. Learning with multiple stacking for named entity recognition. In Dan Roth and Antal van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 191–194. Taipei, Taiwan, 2002.

Dekai Wu, Grace Ngai, Marine Carpuat, Jeppe Larsen, and Yongsheng Yang. Boosting for named entity recognition. In Dan Roth and Antal van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 195–198. Taipei, Taiwan, 2002.

Dekai Wu, Grace Ngai, and Marine Carpuat. N-fold templated piped correction. In *First International Joint Conference on Natural Language Processing (IJCNLP-2004)*, pages 632–637. Hainan Island, China, March 2004.

Dekai Wu, Grace Ngai, and Marine Carpuat. Raising the bar: Stacked conservative error correction beyond boosting. In *Fourth International Conference on Language Resources and Evaluation (LREC-2004)*. Lisbon, May 2004.