Exploiting Diversity in Natural Language Processing: Combining Parsers

John C. Henderson and Eric Brill
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
{jhndrsn,brill}@cs.jhu.edu

Abstract

Three state-of-the-art statistical parsers are combined to produce more accurate parses, as well as new bounds on achievable Treebank parsing accuracy. Two general approaches are presented and two combination techniques are described for each approach. Both parametric and non-parametric models are explored. The resulting parsers surpass the best previously published performance results for the Penn Treebank.

1 Introduction

The natural language processing community is in the strong position of having many available approaches to solving some of its most fundamental problems. The machine learning community has been in a similar situation and has studied the combination of multiple classifiers (Wolpert, 1992; Heath et al., 1996). Their theoretical finding is simply stated: classification error rate decreases toward the noise rate exponentially in the number of independent, accurate classifiers. The theory has also been validated empirically.

Recently, combination techniques have been investigated for part of speech tagging with positive results (van Halteren et al., 1998; Brill and Wu, 1998). In both cases the investigators were able to achieve significant improvements over the previous best tagging results. Similar advances have been made in machine translation (Frederking and Nirenburg, 1994), speech recognition (Fiscus, 1997) and named entity recognition (Borthwick et al., 1998).

The corpus-based statistical parsing community has many fast and accurate automated parsing systems, including systems produced by Collins (1997), Charniak (1997) and Ratnaparkhi (1997). These three parsers have given the best reported parsing results on the Penn Treebank Wall Street Journal corpus (Marcus et al., 1993). We used these three parsers to explore parser combination techniques.

2 Techniques for Combining Parsers

2.1 Parse Hybridization

We are interested in combining the substructures of the input parses to produce a better parse. We call this approach *parse hybridization*. The substructures that are unanimously hypothesized by the parsers should be preserved after combination, and the combination technique should not foolishly create substructures for which there is no supporting evidence. These two principles guide experimentation in this framework, and together with the evaluation measures help us decide which specific type of substructure to combine.

The precision and recall measures (described in more detail in Section 3) used in evaluating Treebank parsing treat each constituent as a separate entity, a minimal unit of correctness. Since our goal is to perform well under these measures we will similarly treat constituents as the minimal substructures for combination.

2.1.1 Constituent Voting

One hybridization strategy is to let the parsers vote on constituents' membership in the hypothesized set. If enough parsers suggest that a particular constituent belongs in the parse, we include it. We call this technique *constituent voting*. We include a constituent in our hypothesized parse if it appears in the output of a majority of the parsers. In our particular case the majority requires the agreement of only two parsers because we have only three. This technique has the advantage of requiring no training, but it has the disadvantage of treating all parsers equally even though they may have differing accuracies or may specialize in modeling different phenomena.

2.1.2 Naïve Bayes

Another technique for parse hybridization is to use a naïve Bayes classifier to determine which constituents to include in the parse. The development of a naïve Bayes classifier involves learning how much each parser should be trusted for the decisions it makes. Our original hope in combining these parsers is that their errors are independently distributed. This is equivalent to the assumption used in probability estimation for naïve Bayes classifiers, namely that the attribute values are conditionally independent when the target value is given. For this reason, naïve Bayes classifiers are well-matched to this problem.

In Equations 1 through 3 we develop the model for constructing our parse using naïve Bayes classification. $\mathcal{C}$ is the union of the sets of constituents suggested by the parsers. $\pi(c)$ is a binary function returning t (for *true*) precisely when the constituent $c \in \mathcal{C}$ should be included in the hypothesis. $M_i(c)$ is a binary function returning t when parser i (from among the k parsers) suggests constituent c should be in the parse. The hypothesized parse is then the set of constituents that are likely ($P > 0.5$) to be in the parse according to this model.

$$\begin{aligned}
&\operatorname*{argmax}_{\pi(c)} P(\pi(c)|M_1(c)\dots M_k(c)) \\
&= \operatorname*{argmax}_{\pi(c)} \frac{P(M_1(c)\dots M_k(c)|\pi(c))P(\pi(c))}{P(M_1(c)\dots M_k(c))} \quad (1)\\
&= \operatorname*{argmax}_{\pi(c)} P(\pi(c)) \prod_{i=1}^{k} \frac{P(M_i(c)|\pi(c))}{P(M_i(c))} \quad (2)\\
&= \operatorname*{argmax}_{\pi(c)} P(\pi(c)) \prod_{i=1}^{k} P(M_i(c)|\pi(c)) \quad (3)
\end{aligned}$$

The estimation of the probabilities in the model is carried out as shown in Equation 4. Here $N(\cdot)$ counts the number of hypothesized constituents in the development set that match the binary predicate specified as an argument.

$$\begin{aligned}
&P(\pi(c)=t) \prod_{i=1}^{k} P(M_i(c)|\pi(c)=t) \\
&= \frac{N(\pi(c)=t)}{|\mathcal{C}|} \prod_{i=1}^{k} \frac{N(M_i(c),\pi(c)=t)}{N(\pi(c)=t)} \quad (4)
\end{aligned}$$

2.1.3 Lemma: No Crossing Brackets

Under certain conditions the constituent voting and naïve Bayes constituent combination techniques are guaranteed to produce sets of constituents with no crossing brackets. There are simply not enough votes remaining to allow any of the crossing structures to enter the hypothesized constituent set.

Lemma: If the number of votes required by constituent voting is greater than half of the parsers under consideration the resulting structure has no crossing constituents.

Proof: Assume a pair of crossing constituents appears in the output of the constituent voting technique using k parsers. Call the crossing constituents A and B. A receives a votes, and B receives b votes. Each of the constituents must have received at least $\lceil \frac{k+1}{2} \rceil$ votes from the k parsers, so $a \geq \lceil \frac{k+1}{2} \rceil$ and $b \geq \lceil \frac{k+1}{2} \rceil$. Let $s = a + b$. None of the parsers produce parses with crossing brackets, so none of them votes for both of the assumed constituents. Hence, $s \leq k$. But by addition of the votes on the two parses, $s \geq 2\lceil \frac{k+1}{2} \rceil > k$, a contradiction. ■

Similarly, when the naïve Bayes classifier is configured such that the constituents require estimated probabilities strictly larger than 0.5 to be accepted, there is not enough probability mass remaining on crossing brackets for them to be included in the hypothesis.

2.2 Parser Switching

In general, the lemma of the previous section does not ensure that all the productions in the combined parse are found in the grammars of the member parsers. There is a guarantee of no crossing brackets but there is no guarantee that a constituent in the tree has the same children as it had in any of the three original parses. One can trivially create situations in which strictly binary-branching trees are combined to create a tree with only the root node and the terminal nodes, a completely flat structure.

This drastic tree manipulation is not appropriate for situations in which we want to assign particular structures to sentences. For example, we may have semantic information (e.g. database query operations) associated with the productions in a grammar. If the parse contains productions from outside our grammar the machine has no direct method for handling them (e.g. the resulting database query may be syntactically malformed).

We have developed a general approach for combining parsers when preserving the entire structure of a parse tree is important. The combining algorithm is presented with the candidate parses and asked to choose which one is best. The combining technique must act as a multi-position switch indicating which parser should be trusted for the particular sentence. We call this approach *parser switching*. Once again we present both a non-parametric and a parametric technique for this task.

2.2.1 Similarity Switching

First we present the non-parametric version of parser switching, *similarity switching*:

1. From each candidate parse, π_i, for a sentence, create the constituent set S_i by converting each constituent into its tuple representation.
2. The score for π_i is $\sum_{j \neq i} |S_j \cap S_i|$, where j ranges over the candidate parses for the sentence.
3. Switch to (use) the parser with the highest score for the sentence. Ties are broken arbitrarily.

The intuition for this technique is that we can measure a similarity between parses by counting the constituents they have in common. We pick the parse that is most similar to the other parses by choosing the one with the highest sum of pairwise similarities. This is the parse that is closest to the centroid of the observed parses under the similarity metric.

2.2.2 Naïve Bayes

The probabilistic version of this procedure is straightforward. We once again assume independence among our various member parsers. Furthermore, we know one of the original parses will be the hypothesized parse, so the direct method of determining which one is best is to compute the probability of each of the candidate parses using the probabilistic model we developed in Section 2.1. We model each parse as the decisions made to create it, and model those decisions as independent events. Each decision determines the inclusion or exclusion of a candidate constituent. The set of candidate constituents comes from the union of all the constituents suggested by the member parsers. This is summarized in Equation 5. The computation of $P(\pi_i(c)|M_1 \dots M_k(c))$ has been sketched before in Equations 1 through 4. In this case we are interested in finding the maximum probability parse, π_i, and M_i is the set of relevant (binary) parsing decisions made by parser i. π_i is a parse selected from among the outputs of the individual parsers. It is chosen such that the decisions it made in including or excluding constituents are most probable under the models for *all* of the parsers.

$$\begin{aligned} & \operatorname*{argmax}_{\pi_i} P(\pi_i|M_1 \dots M_k) \\ = & \operatorname*{argmax}_{\pi_i} \prod_c P(\pi_i(c)|M_1(c) \dots M_k(c)) \quad (5) \end{aligned}$$

3 Experiments

The three parsers were trained and tuned by their creators on various sections of the WSJ portion of the Penn Treebank, leaving only sections 22 and 23 completely untouched during the development of any of the parsers. We used section 23 as the development set for our combining techniques, and section 22 only for final testing. The development set contained 44088 constituents in 2416 sentences and the test set contained 30691 constituents in 1699 sentences. A sentence was withheld from section 22 because its extreme length was troublesome for a couple of the parsers.[1]

The standard measures for evaluating Penn Treebank parsing performance are precision and recall of the predicted constituents. Each parse is converted into a set of constituents represented as a tuples: (label, start, end). The set is then compared with the set generated from the Penn Treebank parse to determine the precision and recall. **Precision** is the portion of hypothesized constituents that are correct and **recall** is the portion of the Treebank constituents that are hypothesized.

For our experiments we also report the mean of precision and recall, which we denote by $(P + R)/2$ and F-measure. F-measure is the harmonic mean of precision and recall, $2PR/(P + R)$. It is closer to the smaller value of precision and recall when there is a large skew in their values.

We performed three experiments to evaluate our techniques. The first shows how constituent features and context do not help in deciding which parser to trust. We then show that the combining techniques presented above give better parsing accuracy than any of the individual parsers. Finally we show the combining techniques degrade very little when a poor parser is added to the set.

3.1 Context

It is possible one could produce better models by introducing features describing constituents and their contexts because one parser could be much better than the majority of the others in particular situations. For example, one parser could be more accurate at predicting noun phrases than the other parsers. None of the models we have presented utilize features associated with a particular constituent (i.e. the label, span, parent label, etc.) to influence parser preference. This is not an oversight. Features and context were initially introduced into the models, but they refused to offer any gains in performance. While we cannot prove there are no such useful features on which one should condition trust, we can give some insight into why the features we explored offered no gain.

Because we are working with only three parsers, the only situation in which context will help us is when it can indicate we should choose to believe a single parser that disagrees with the majority hypothesis instead of the majority hypothesis itself. This is the only important case, because otherwise the simple majority combining technique would pick the correct constituent. One side of the decision making process is when we choose to believe a constituent should be in the parse, even though only one parser suggests it. We call such a constituent an *isolated constituent.* If we were working with more than three parsers we could investigate *minority constituents*, those constituents that are suggested by at least one parser, but which the majority of the parsers do not suggest.

Adding the isolated constituents to our hypothesis parse could increase our expected recall, but in the cases we investigated it would invariably hurt our precision more than we would gain on recall.

[1]The sentence in question was more than 100 words in length and included nested quotes and parenthetical expressions.

Consider for a set of constituents the ***isolated constituent precision*** parser metric, the portion of isolated constituents that are correctly hypothesized. When this metric is less than 0.5, we expect to incur more errors[2] than we will remove by adding those constituents to the parse.

We show the results of three of the experiments we conducted to measure isolated constituent precision under various partitioning schemes. In Table 1 we see with very few exceptions that the isolated constituent precision is less than 0.5 when we use the constituent label as a feature. The counts represent portions of the approximately 44000 constituents hypothesized by the parsers in the development set. In the cases where isolated constituent precision is larger than 0.5 the affected portion of the hypotheses is negligible.

Similarly Figures 1 and 2 show how the isolated constituent precision varies by sentence length and the size of the span of the hypothesized constituent. In each figure the upper graph shows the isolated constituent precision and the bottom graph shows the corresponding number of hypothesized constituents. Again we notice that the isolated constituent precision is larger than 0.5 only in those partitions that contain very few samples. From this we see that a finer-grained model for parser combination, at least for the features we have examined, will not give us any additional power.

3.2 Performance Testing

The results in Table 2 were achieved on the development set. The first two rows of the table are baselines. The first row represents the average accuracy of the three parsers we combine. The second row is the accuracy of the best of the three parsers.[3] The next two rows are results of oracle experiments. The parser switching oracle is the upper bound on the accuracy that can be achieved on this set in the parser switching framework. It is the performance we could achieve if an omniscient observer told us which parser to pick for each of the sentences. The maximum precision row is the upper bound on accuracy if we could pick exactly the correct constituents from among the constituents suggested by the three parsers. Another way to interpret this is that less than 5% of the correct constituents are missing from the hypotheses generated by the union of the three parsers. The maximum precision oracle is an upper bound on the possible gain we can achieve by parse hybridization.

[2]This is in absolute terms, total errors being the sum of precision errors and recall errors.

[3]The identity of this parser is not given, nor is the identity disclosed for the results of any of the individual parsers. We do not aim to compare the performance of the individual parsers, nor do we want to bias further research by giving the individual parser results for the test set.

Parser	Sentences	%
Parser 1	279	16
Parser 2	216	13
Parser 3	1204	71

Table 4: Bayes Switching Parser Usage

We do not show the numbers for the Bayes models in Table 2 because the parameters involved were established using this set. The precision and recall of similarity switching and constituent voting are both significantly better than the best individual parser, and constituent voting is significantly better than parser switching in precision.[4] Constituent voting gives the highest accuracy for parsing the Penn Treebank reported to date.

Table 3 contains the results for evaluating our systems on the test set (section 22). All of these systems were run on data that was not seen during their development. The difference in precision between similarity and Bayes switching techniques is significant, but the difference in recall is not. This is the first set that gives us a fair evaluation of the Bayes models, and the Bayes switching model performs significantly better than its non-parametric counterpart. The constituent voting and naïve Bayes techniques are equivalent because the parameters learned in the training set did not sufficiently discriminate between the three parsers.

Table 4 shows how much the Bayes switching technique uses each of the parsers on the test set. Parser 3, the most accurate parser, was chosen 71% of the time, and Parser 1, the least accurate parser was chosen 16% of the time. Ties are rare in Bayes switching because the models are fine-grained – many estimated probabilities are involved in each decision.

3.3 Robustness Testing

In the interest of testing the robustness of these combining techniques, we added a fourth, simple nonlexicalized PCFG parser. The PCFG was trained from the same sections of the Penn Treebank as the other three parsers. It was then tested on section 22 of the Treebank in conjunction with the other parsers.

The results of this experiment can be seen in Table 5. The entries in this table can be compared with those of Table 3 to see how the performance of the combining techniques degrades in the presence of an inferior parser. As seen by the drop in average individual parser performance baseline, the introduced parser does not perform very well. The average individual parser accuracy was reduced by more than 5% when we added this new parser, but the preci-

[4]All significance claims are made based on a binomial hypothesis test of equality with an $\alpha < 0.01$ confidence level.

Constituent Label	Parser1		Parser2		Parser3	
	count	Precision	count	Precision	count	Precision
ADJP	132	28.78	215	21.86	173	34.10
ADVP	150	25.33	129	21.70	102	31.37
CONJP	2	50.00	8	37.50	3	0.00
FRAG	51	3.92	29	27.58	11	9.09
INTJ	3	66.66	1	100.00	2	50.00
LST	0	NA	0	NA	0	NA
NAC	0	NA	13	53.84	7	14.28
NP	1489	21.08	1550	18.38	1178	27.33
NX	7	85.71	9	22.22	3	0.00
PP	732	23.63	643	20.06	503	27.83
PRN	20	55.00	33	54.54	38	15.78
PRT	12	16.66	20	40.00	16	37.50
QP	21	38.09	34	44.11	76	14.47
RRC	1	0.00	1	0.00	2	0.00
S	757	13.73	482	23.65	434	38.94
SBAR	331	11.78	196	23.97	178	34.83
SBARQ	0	NA	6	16.66	3	0.00
SINV	3	66.66	11	81.81	13	30.76
SQ	2	0	11	18.18	3	33.33
UCP	6	16.66	12	8.33	8	12.50
VP	868	13.36	630	24.12	477	35.42
WHADJP	0	NA	0	NA	1	0.00
WHADVP	2	100.00	5	40.00	1	100.00
WHNP	33	33.33	8	25.00	17	58.82
WHPP	0	NA	0	NA	2	100.00
X	0	NA	2	100.00	1	0.00

Table 1: Isolated Constituent Precision By Constituent Label

Reference / System	P	R	(P+R)/2	F
Average Individual Parser	87.14	86.91	87.02	87.02
Best Individual Parser	88.73	88.54	88.63	88.63
Parser Switching Oracle	93.12	92.84	92.98	92.98
Maximum Precision Oracle	100.00	95.41	97.70	97.65
Similarity Switching	89.50	89.88	89.69	89.69
Constituent Voting	92.09	89.18	90.64	90.61

Table 2: Summary of Development Set Performance

Reference / System	P	R	(P+R)/2	F
Average Individual Parser	87.61	87.83	87.72	87.72
Best Individual Parser	89.61	89.73	89.67	89.67
Parser Switching Oracle	93.78	93.87	93.82	93.82
Maximum Precision Oracle	100.00	95.91	97.95	97.91
Similarity Switching	90.04	90.81	90.43	90.43
Bayes Switching	90.78	90.70	90.74	90.74
Constituent Voting	92.42	90.10	91.26	91.25
Naïve Bayes	92.42	90.10	91.26	91.25

Table 3: Test Set Results

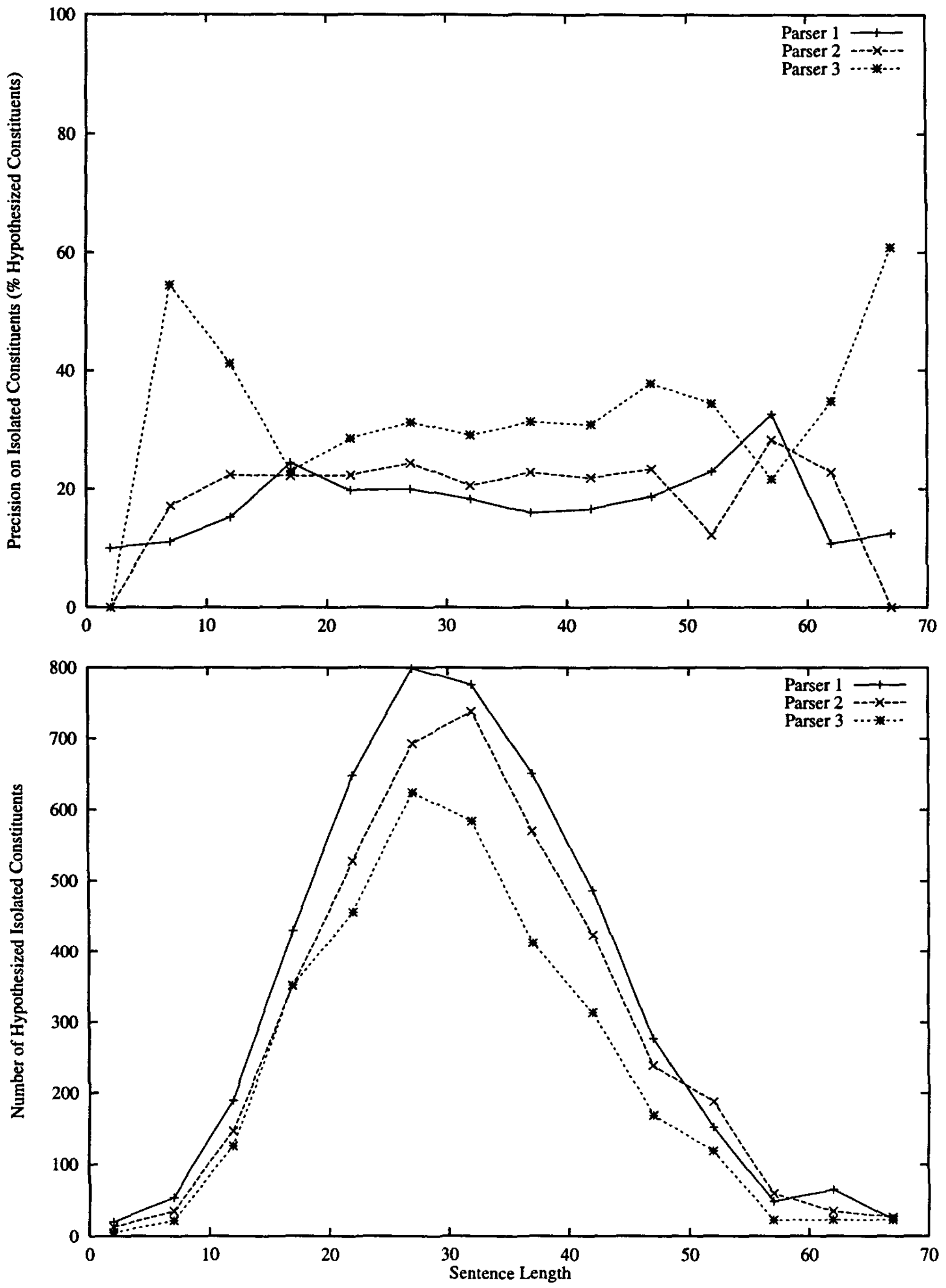

Figure 1: Isolated Constituent Precision and Sentence Length

sion of the constituent voting technique was the only result that decreased significantly. The Bayes models were able to achieve significantly higher precision than their non-parametric counterparts. We see from these results that the behavior of the parametric techniques are robust in the presence of a poor parser. Surprisingly, the non-parametric switching technique also exhibited robust behaviour in this situation.

4 Conclusion

We have presented two general approaches to studying parser combination: parser switching and parse hybridization. For each experiment we gave an non-

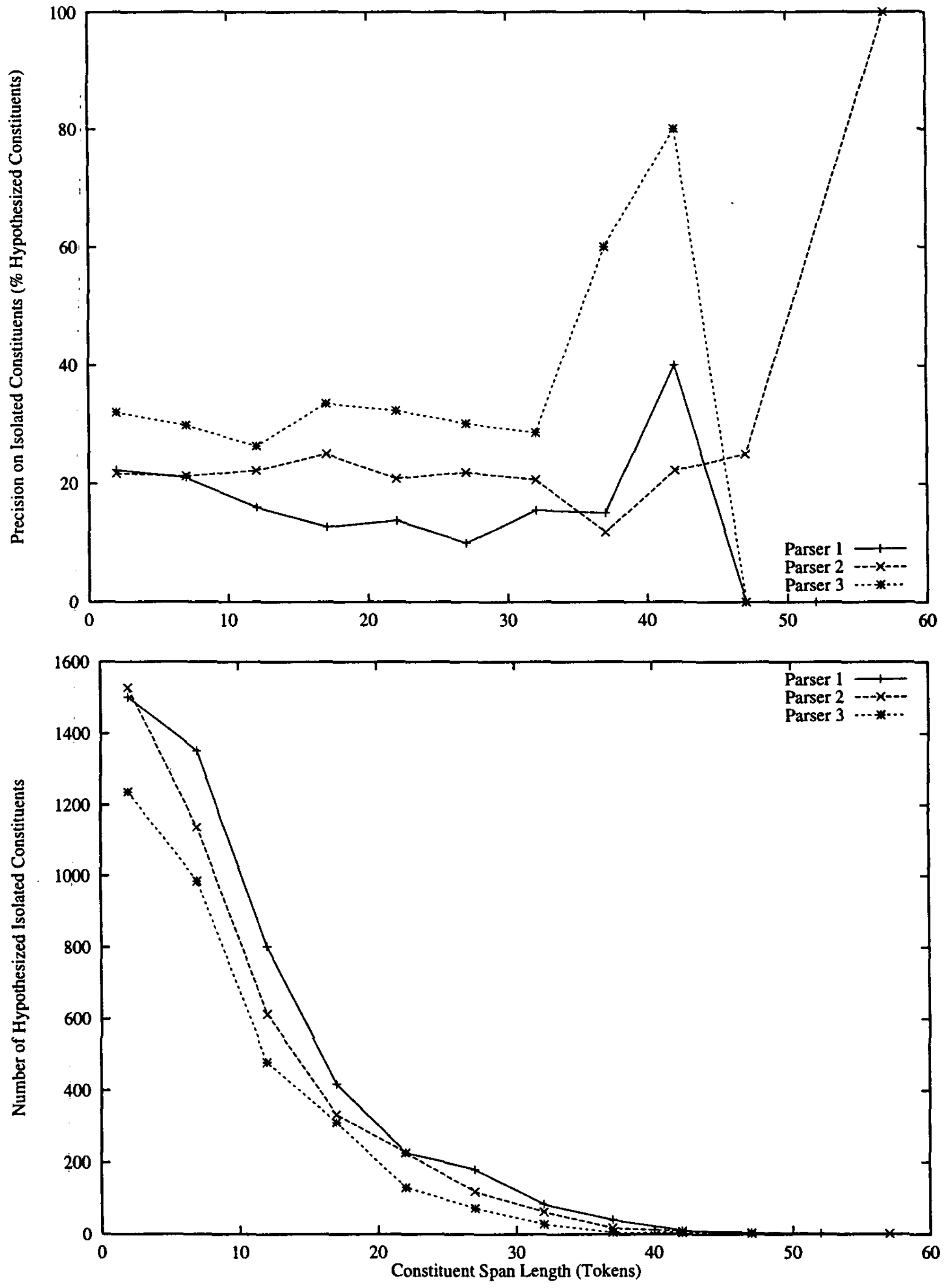

Figure 2: Isolated Constituent Precision and Span Length

parametric and a parametric technique for combining parsers. All four of the techniques studied result in parsing systems that perform better than any previously reported. Both of the switching techniques, as well as the parametric hybridization technique were also shown to be robust when a poor parser was introduced into the experiments. Through parser combination we have reduced the precision error rate by 30% and the recall error rate by 6% compared to the best previously published result.

Combining multiple highly-accurate independent parsers yields promising results. We plan to explore more powerful techniques for exploiting the diversity of parsing methods.

Reference / System	P	R	(P+R)/2	F
Average Individual Parser	84.55	80.91	82.73	82.69
Best Individual Parser	89.61	89.73	89.67	89.67
Parser Switching Oracle	93.92	93.88	93.90	93.90
Maximum Precision Oracle	100.00	96.66	98.33	98.30
Similarity Switching	89.90	90.89	90.40	90.39
Bayes Switching	90.94	90.70	90.82	90.82
Constituent Voting	89.78	91.80	90.79	90.78
Naïve Bayes	92.42	90.10	91.26	91.25

Table 5: Robustness Test Results

5 Acknowledgements

We would like to thank Eugene Charniak, Michael Collins, and Adwait Ratnaparkhi for enabling all of this research by providing us with their parsers and helpful comments.

This work was funded by NSF grant IRI-9502312. Both authors are members of the Center for Language and Speech Processing at Johns Hopkins University.

References

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In Eugene Charniak, editor, *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 152–160, Montreal.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical combination. In *Proceedings of the 17th International Conference on Computational Linguistics.*

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, volume 35, Madrid.

Jonathan G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *EuroSpeech 1997 Proceedings*, volume 4, pages 1895–1898.

Robert Frederking and Sergei Nirenburg. 1994. Three heads are better than one. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 95–100, Stuttgart, Germany.

David Heath, Simon Kasif, and Steven Salzberg. 1996. Committees of decision trees. In B. Gorayska and J. Mey, editors, *Cognitive Technology: In Search of a Humane Interface*, pages 305–317. Elsevier Science B.V., Amsterdam.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Second Conference on Empirical Methods in Natural Language Processing*, Providence, R.I.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the 17th International Conference on Computational Linguistics*, Montreal.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.