Comparing and Combining Finite-State and Context-Free Parsers

Kristy Hollingshead and **Seeger Fisher** and **Brian Roark**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
Beaverton, Oregon, 97006
{hollingk,fishers,roark}@cslu.ogi.edu

Abstract

In this paper, we look at comparing high-accuracy context-free parsers with high-accuracy finite-state (shallow) parsers on several shallow parsing tasks. We show that previously reported comparisons greatly under-estimated the performance of context-free parsers for these tasks. We also demonstrate that context-free parsers can train effectively on relatively little training data, and are more robust to domain shift for shallow parsing tasks than has been previously reported. Finally, we establish that combining the output of context-free and finite-state parsers gives much higher results than the previous-best published results, on several common tasks. While the efficiency benefit of finite-state models is inarguable, the results presented here show that the corresponding cost in accuracy is higher than previously thought.

1 Introduction

Finite-state parsing (also called chunking or shallow parsing) has typically been motivated as a fast first-pass for – or approximation to – more expensive context-free parsing (Abney, 1991; Ramshaw and Marcus, 1995; Abney, 1996). For many very-large-scale natural language processing tasks (e.g. open-domain question answering from the web), context-free parsing may be too expensive, whereas finite-state parsing is many orders of magnitude faster and can also provide very useful syntactic annotations for large amounts of text. For this reason, finite-state parsing (hereafter referred to as shallow parsing) has received increasing attention in recent years.

In addition to the clear efficiency benefit of shallow parsing, Li and Roth (2001) have further claimed both an accuracy and a robustness benefit versus context-free parsing. The output of a context-free parser, such as that of Collins (1997) or Charniak (2000), can be transformed into a sequence of shallow constituents for comparison with the output of a shallow parser. Li and Roth demonstrated that their shallow parser, trained to label shallow constituents along the lines of the well-known CoNLL-2000 task (Sang and Buchholz, 2000), outperformed the Collins parser in correctly identifying these constituents in the Penn Wall Street Journal (WSJ) Treebank (Marcus et al., 1993). They argued that their superior performance was due to optimizing directly for the local sequence labeling objective, rather than for obtaining a hierarchical analysis over the entire string. They further showed that their shallow parser trained on the Penn WSJ Treebank did a far better job of annotating out-of-domain sentences (e.g. conversational speech) than the Collins parser.

This paper re-examines the comparison of shallow parsers with context-free parsers, beginning with a critical examination of how their outputs are compared. We demonstrate that changes to the conversion routine, which take into account differences between the original treebank trees and the trees output by context-free parsers, eliminate the previously-reported accuracy differences. Second, we show that a convention that is widely accepted for evaluation of context-free parses – ignoring punctuation when setting the span of a constituent – results in improved shallow parsing performance by certain context-free parsers across a variety of shallow parsing tasks. We also demonstrate that context-free parsers perform competitively when applied to out-of-domain data. Finally, we show that large improvements can be obtained in several shallow parsing tasks by using simple strategies to incorporate context-free parser output into shallow parsing models. Our results demonstrate that a rich context-free

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 787–794, Vancouver, October 2005.

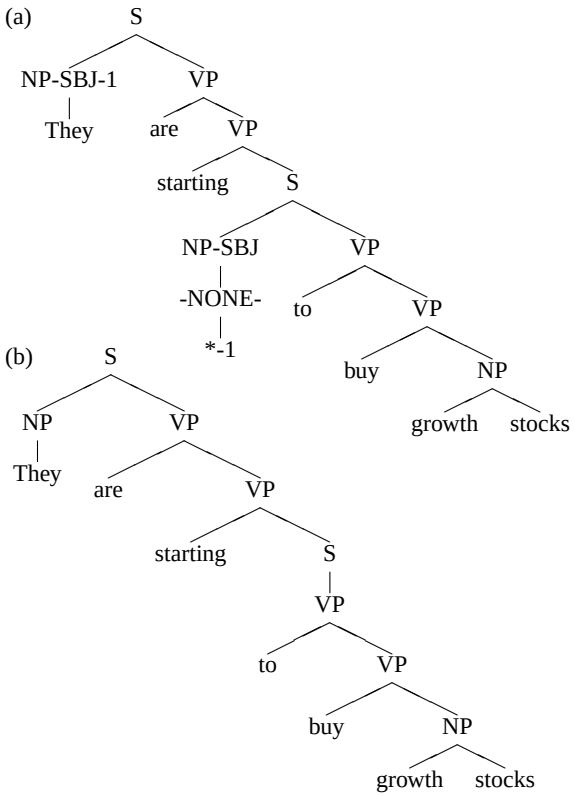
parsing model is, time permitting, worth applying, even if only shallow parsing output is needed. In addition, our best results, which greatly improve on the previous-best published results on several tasks, shed light on how much accuracy is sacrificed in shallow parsing to get finite-state efficiency.

2 Evaluating Heterogeneous Parser Output

Two commonly reported shallow parsing tasks are Noun-Phrase (NP) Chunking (Ramshaw and Marcus, 1995) and the CoNLL-2000 Chunking task (Sang and Buchholz, 2000), which extends the NP-Chunking task to recognition of 11 phrase types[1] annotated in the Penn Treebank. Reference shallow parses for this latter task were derived from treebank trees via a conversion script known as `chunklink`[2]. We follow Li and Roth (2001) in using `chunklink` to also convert trees output by a context-free parser into a flat representation of shallow constituents. Figure 1(a) shows a Penn Treebank tree and Figure 1(c) its corresponding shallow parse constituents, according to the CoNLL-2000 guidelines. Note that consecutive verb phrase (VP) nodes result in a single VP shallow constituent.

Just as the original treebank trees are converted for training shallow parsers, they are also typically modified for training context-free parsers. This modification includes removal of empty nodes (nodes tagged with "-NONE-" in the treebank), and removal of function tags on non-terminals; e.g., NP-SBJ (subject NP) and NP-TMP (temporal NP) are both mapped to NP. The output of the context-free parser is, of course, in the same format as the training input, so empty nodes and function tags are not present. This type of modified tree is what is shown in Figure 1(b); note that the original treebank tree, shown in Figure 1(a), had an empty subject NP in the embedded clause which has been removed for the modified tree.

To compare the output of their shallow parser with the output of the well-known Collins (1997) parser, Li and Roth applied the `chunklink` conversion script to extract the shallow constituents from the output of the Collins parser on WSJ section 00. Unfortunately, the script was built to be applied to the original treebank trees, complete with empty nodes, which are not present in the output of the Collins parser, or any well-known context-free parser. The `chunklink` script searches for empty nodes in the parse tree to perform some of its operations. In particular, any S node that contains an empty subject NP and a VP is reduced to just a VP node, and then combined with any immediately-preceding VP nodes to create a single VP constituent. If the S node does not contain an empty subject NP, as in Figure 1(b), the `chunklink` script creates two VP constituents: [VP are starting] [VP to buy], which in this case results in a bracketing error. However, it is a simple matter to insert an empty subject NP into unary S→VP productions so that these nodes are processed correctly by the script.



(c) [NP They] [VP are starting to buy] [NP growth stocks]

Figure 1: (a) Penn WSJ treebank tree, (b) modified treebank tree, and (c) CoNLL-2000 style shallow bracketing, all of the same string.

Various conventions have become standard in evaluating parser output over the past decade. Perhaps the most widely accepted convention is that of ignoring punctuation for the purposes of assigning constituent span, under the perspective that, fun-

[1] These include: ADJP, ADVP, CONJP, INTJ, LST, NP, PP, PRT, SBAR, UCP and VP. Anything not in one of these base phrases is designated as "outside".

[2] Downloaded from http://ilk.kub.nl/∼sabine/chunklink/.

System	Phrase Type	Evaluation Scenario (a)	(b)	(c)
"Modified" Truth	All	98.37	99.72	99.72
	VP	92.14	98.70	98.70
Li and Roth (2001)	All	94.64	-	-
	VP	95.28	-	-
Collins (1997)	All	92.16	93.42	94.28
	VP	88.15	94.31	94.42
Charniak (2000)	All	93.88	95.15	95.32
	VP	88.92	95.11	95.19

Table 1: F-measure shallow bracketing accuracy under three different evaluation scenarios: (a) baseline, used in Li and Roth (2001), with original `chunklink` script converting treebank trees and context-free parser output; (b) same as (a), except that empty subject NPs are inserted into every unary S→VP production; and (c) same as (b), except that punctuation is ignored for setting constituent span. Results for Li and Roth are reported from their paper. The Collins parser is provided with part-of-speech tags output by the Brill tagger (Brill, 1995).

damentally, constituents are groupings of words. Interestingly, this convention was not followed in the CoNLL-2000 task (Sang and Buchholz, 2000), which as we will see has a variable effect on context-free parsers, presumably depending on the degree to which punctuation is moved in training.

2.1 Evaluation Analysis

To determine the effects of the conversion routine and different evaluation conventions, we compare the performance of several different models on one of the tasks presented in Li and Roth (2001). For this task, which we label the *Li & Roth task*, sections 2-21 of the Penn WSJ Treebank are used as training data, section 24 is held out, and section 00 is for evaluation.

For all trials in this paper, we report F-measure labeled bracketing accuracy, which is the harmonic mean of the labeled precision (P) and labeled recall (R), as they are defined in the widely used PARSEVAL metrics; i.e. the F-measure accuracy is $\frac{2PR}{P+R}$.

Table 1 shows baseline results for the Li and Roth[3] shallow parser, two well-known, high-accuracy context-free parsers, and the reference (true) parses after being modified as described above (by removing empty nodes and function tags). Evaluation scenario (a) in Table 1 corresponds to what was used in Li and Roth (2001) following CoNLL-2000 guidelines, with the original `chunklink` script used to transform the context-free parser output into shallow constituents. We can see from the performance of the modified truth in this scenario that there are serious problems with this conversion, due to the way in which it handles unary S→VP productions. If we deterministically insert empty subject NP nodes for all such unary productions prior to the use of the `chunklink` script, which we do in evaluation scenario (b) of Table 1, this repairs the bulk of the errors. Some small number of errors remain, due largely to the fact that if the S node has been annotated with a function tag (e.g. S-PRP, S-PRD, S-CLR), then `chunklink` will not perform its reduction operation on that node. However, for our purposes, this insertion repair sufficiently corrects the error to perform meaningful comparisons. Finally, evaluation scenario (c) follows the context-free parsing evaluation convention of ignoring punctuation when assigning constituent span. This affects some parsers more than others, depending on how the parser treats punctuation internally; for example, Bikel (2004) documents that the Collins parser raises punctuation nodes within the parse tree. Since ignoring punctuation cannot hurt performance, only improve it, even the smallest of these differences are statistically significant.

Note that after inserting empty nodes and ignoring punctuation, the accuracy advantage of Li and Roth over Collins is reduced to a dead heat. Of the two parsers we evaluated, the Charniak (2000) parser gave the best performance, which is consistent with its higher reported performance on the context-free parsing task versus other context-free parsers. Collins (2000) reported a reranking model that improved his parser output to roughly the same level of accuracy as Charniak (2000), and Charniak and Johnson (2005) report an improvement using reranking over Charniak (2000). For the purposes of this paper, we needed an available parser that was (a) trainable on different subsets of the data to be applied to various tasks; and (b) capable of producing n-best candidates, for potential combination with a shallow parser. Both the Bikel (2004) imple-

[3]We were unable to obtain the exact model used in Li and Roth (2001), and so we use their reported results here. Note that they used reference part-of-speech (POS) tags for their results on this task. All other results reported in this paper, unless otherwise noted, were obtained using Brill-tagger POS tags.

System	NP-Chunking	CoNLL-2000	Li & Roth task
SPRep averaged perceptron	94.21	93.54	95.12
Kudo and Matsumoto (2001)	94.22	93.91	-
Sha and Pereira (2003) CRF	94.38	-	-
Voted perceptron	94.09	-	-
Zhang et al. (2002)	-	94.17	-
Li and Roth (2001)	-	93.02	94.64

Table 2: Baseline results on three shallow parsing tasks: the NP-Chunking task (Ramshaw and Marcus, 1995); the CoNLL-2000 Chunking task (Sang and Buchholz, 2000); and the Li & Roth task (Li and Roth, 2001), which is the same as CoNLL-2000 but with more training data and a different test section. The results reported in this table include the best published results on each of these tasks.

mentation of the Collins parser and the n-best version of the Charniak (2000) parser, documented in Charniak and Johnson (2005), fit the requirements. Since we observed higher accuracy from the Charniak parser, from this point forward we report just Charniak parser results[4].

2.2 Shallow Parser

In addition to the trainable n-best context-free parser from Charniak (2000), we needed a trainable shallow parser to apply to the variety of tasks we were interested in investigating. To this end, we replicated the NP-chunker described in Sha and Pereira (2003) and trained it as either an NP-chunker or with the tagset extended to classify all 11 phrase types included in the CoNLL-2000 task (Sang and Buchholz, 2000). Our shallow parser uses exactly the feature set delineated by Sha and Pereira, and performs the decoding process using a Viterbi search with a second-order Markov assumption as they described. These features include unigram and bigram words up to two positions to either side of the current word; unigram, bigram, and trigram part-of-speech (POS) tags up to two positions to either side of the current word; and unigram, bigram, and trigram shallow constituent tags. We use the averaged perceptron algorithm, as presented in Collins (2002), to train the parser. See (Sha and Pereira, 2003) for more details on this approach.

To demonstrate the competitiveness of our baseline shallow parser, which we label the *SPRep averaged perceptron*, Table 2 shows results on three different shallow parsing tasks. The NP-Chunking task, originally introduced in Ramshaw and Marcus (1995) and also described in (Collins, 2002; Sha and Pereira, 2003), brackets just base NP constituents[5]. The CoNLL-2000 task, introduced as a shared task at the CoNLL workshop in 2000 (Sang and Buchholz, 2000), extends the NP-Chunking task to label 11 different base phrase constituents. For both of these tasks, the training set was sections 15-18 of the Penn WSJ Treebank and the test set was section 20. We follow Collins (2002) and Sha and Pereira (2003) in using section 21 as a heldout set. The third task, introduced by Li and Roth (2001), performs the same labeling as in the CoNLL-2000 task, but with more training data and different testing sets: training was WSJ sections 2-21 and test was section 00. We used section 24 as a heldout set; this section is often used as heldout for training context-free parsers.

Training and testing data for the CoNLL-2000 task is available online[6]. For the heldout sets for each of these tasks, as well as for all data sets needed for the Li & Roth task, reference shallow parses were generated using the `chunklink` script on the original treebank trees. All data was tagged with the Brill POS tagger (Brill, 1995) after the `chunklink` conversion. We verified that using this method on the original treebank trees in sections 15-18 and 20 generated data that is identical to the CoNLL-2000 data sets online. Replacing the POS tags in the input text with Brill POS tags before the

[4]The parser is available for research purposes at ftp://ftp.cs.brown.edu/pub/nlparser/ and can be run in n-best mode. The one-best performance of the parser is the same as what was presented in Charniak (2000).

[5]We follow Sha and Pereira (2003) in deriving the NP constituents from the CoNLL-2000 data sets, by replacing all non-NP shallow tags with the "outside" ("O") tag. They mention that the resulting shallow parse tags are somewhat different than those used by Ramshaw and Marcus (1995), but that they found no significant accuracy differences in training on either set.

[6]Downloaded from the CoNLL-2000 Shared Task website http://www.cnts.ua.ac.be/conll2000/chunking/.

`chunklink` conversion results in slightly different shallow parses.

From Table 2 we can see that our shallow parser is competitive on all three tasks[7]. Sha and Pereira (2003) noted that the difference between their perceptron and CRF results was not significant, and our performance falls between the two, thus replicating their result within noise. Our performance falls 0.6 percentage points below the best published result on the CoNLL-2000 task, and 0.5 percentage points above the performance by Li and Roth (2001) on their task. Overall, ours is a competitive approach for shallow parsing.

3 Experimental Results

3.1 Comparing Finite-State and Context-Free Parsers

The first two rows of Table 3 present a comparison between the SPRep shallow parser and the Charniak (2000) context-free parser detailed in Charniak and Johnson (2005). We can see that the performance of the two models is virtually indistinguishable for all three of these tasks, with or without ignoring of punctuation. As mentioned earlier, we used the version of this parser with improved n-best extraction, as documented in Charniak and Johnson (2005), although without the reranking of the candidates that they also report in that paper. For these trials, we used just the one-best output of that model, which is the same as in Charniak (2000).

Note that the standard training set for context-free parsing (sections 2-21) is only used for the Li & Roth task; for the other two tasks, both the SPRep and the Charniak parsers were trained on sections 15-18, with section 21 as heldout. This demonstrates that the context-free parser, even when trained on a small fraction of the total treebank, is able to learn a competitive model for this task.

3.2 Combining Finite-State and Context-Free Parsers

It is likely true that a context-free parser which has been optimized for global parse accuracy will, on occasion, lose some shallow parse accuracy to satisfy global structure constraints that do not constrain a shallow parser. However, it is also likely true that these longer distance constraints will on occasion enable the context-free parser to better identify the shallow constituent structure. In other words, despite having very similar performance, our shallow parser and the Charniak context-free parser are likely making complementary predictions about the shallow structure that can be exploited for further improvements. In this section, we explore two simple methods for combining the system outputs.

The first combination of the system outputs, which we call *unweighted intersection*, is the simplest kind of 'rovered' system, which restricts the set of shallow parse candidates to the intersection of the sets output by each system, but does not combine the scores. Since the Viterbi search of the SPRep model provides a score for all possible shallow parses, the intersection of the two sets is simply the set of shallow-parse sequences in the 50-best candidates output by the Charniak parser. We then use the SPRep perceptron-model scores to choose from among just these candidates. We converted the 50-best lists returned by the Charniak parser into k-best lists of shallow parses by using `chunklink` to convert each candidate context-free parse into a shallow parse. Many of the context-free parses map to the same shallow parse, so the size of this list is typically much less than 50, with an average of around 7. Each of the unique shallow-parse candidates is given a score by the SPRep perceptron, and the best-scoring candidate is selected. Effectively, we used the Charniak parser's k-best shallow parses to limit the search space for our shallow parser.

The second combination of the system outputs, which we call *weighted intersection*, extends the unweighted intersection by including the scores from the Charniak parser, which are log probabilities. The score for a shallow parse output by the Charniak parser is the log of the sum of the probabilities of all context-free parses mapping to that shallow parse. We normalize across all candidates for a given string, hence these are conditional log probabilities. We multiply these conditional log probabilities by a scaling factor α before adding them to the SPRep perceptron score for a particular candidate. Again, the best-scoring candidate using this composite score is selected from among the shallow

[7] Sha and Pereira (2003) reported the Kudo and Matsumoto (2001) performance on the NP-Chunking task to be 94.39 and to be the best reported result on this task. In the cited paper, however, the result is as reported in our table.

System	NP-Chunking Punctuation Leave	NP-Chunking Punctuation Ignore	CoNLL-2000 Punctuation Leave	CoNLL-2000 Punctuation Ignore	Li & Roth task Punctuation Leave	Li & Roth task Punctuation Ignore
SPRep averaged perceptron	94.21	94.25	93.54	93.70	95.12	95.27
Charniak (2000)	94.17	94.20	93.77	93.92	95.15	95.32
Unweighted intersection	95.13	95.16	94.52	94.64	95.77	95.92
Weighted intersection	95.57	95.58	95.03	95.16	96.20	96.33

Table 3: F-measure shallow bracketing accuracy on three shallow parsing tasks, for the SPRep perceptron shallow parser, the Charniak (2000) context-free parser, and for systems combining the SPRep and Charniak system outputs.

parse candidates output by the Charniak parser. We used the heldout data to empirically estimate an optimal scaling factor for the Charniak scores, which is 15 for all trials reported here. This factor compensates for differences in the dynamic range of the scores of the two parsers.

Both of these intersections are done at test-time, i.e. the models are trained independently. To remain consistent with task-specific training and testing section conventions, the individual models were always trained on the appropriate sections for the given task, i.e. WSJ sections 15-18 for NP-Chunking and the CoNLL-2000 tasks, and sections 2-21 for the Li & Roth task.

Results from these methods of combination are shown in the bottom two rows of Table 3. Even the simple unweighted intersection gives quite large improvements over each of the independent systems for all three tasks. All of these improvements are significant at $p < 0.001$ using the Matched Pair Sentence Segment test (Gillick and Cox, 1989). The weighted intersection gives further improvements over the unweighted intersection for all tasks, and this improvement is also significant at $p < 0.001$, using the same test.

3.3 Robustness to Domain Shift

Our final shallow parsing task was also proposed in Li and Roth (2001). The purpose of this task was to examine the degradation in performance when parsers, trained on one relatively clean domain such as WSJ, are tested on another, mismatched domain such as Switchboard. The systems that are reported in this section are trained on sections 2-21 of the WSJ Treebank, with section 24 as heldout, and tested on section 4 of the Switchboard Treebank. Note that the systems used here are exactly the ones presented for the original Li & Roth task, in Sections 3.1 and 3.2; only the test set has changed, training and heldout sets remain exactly the same, as do the mixing parameters for the weighted intersection. In the trials reported in Li and Roth (2001), both of the evaluated systems were provided with reference POS tags from the Switchboard Treebank. In the current results, we show our SPRep averaged perceptron system provided both with reference POS tags for comparison with the Li and Roth results, and provided with Brill-tagger POS tags for comparison with other systems. Table 4 shows our results for this task. Whereas Li and Roth reported a more marked degradation in performance when using a context-free parser as compared to a shallow parser, we again show virtually indistinguishable performance between our SPRep shallow parser and the Charniak context-free parser. Again, using a weighted combined model gave us large improvements over each independent model, even in this mismatched domain.

System	Punctuation Leave	Punctuation Ignore
Li & Roth (reference tags)	88.47	-
SPRep avg perceptron		
Reference tags	91.37	91.86
Brill tags	87.94	88.42
Charniak (2000)	87.94	88.44
Unweighted intersection	88.66	89.16
Weighted intersection	89.22	89.69

Table 4: Shallow bracketing accuracy of several different systems, trained on sections 2-21 of WSJ Treebank and applied to section 4 of the Switchboard Treebank. Li and Roth (2001) results are as reported in their paper, with reference POS tags rather than Brill-tagger POS tags.

3.4 Reranked n-best List

Just prior to the publication of this paper, we were able to obtain the trained reranker from Charniak

System		WSJ Sect. 00 Punctuation Leave	WSJ Sect. 00 Punctuation Ignore	SWBD Sect. 4 Punctuation Leave	SWBD Sect. 4 Punctuation Ignore
SPRep		95.12	95.27	87.94	88.43
C & J (2005)	one-best	95.15	95.32	87.94	88.44
	reranked	95.81	96.04	88.64	89.17
Weighted intersection		96.32	96.47	89.32	89.80

Table 5: F-measure shallow bracketing accuracy when trained on WSJ sections 2-21 and applied to either WSJ section 00 or SWBD section 4. Systems include our shallow parser (SPRep); the Charniak and Johnson (2005) system (C & J), both initial one-best and reranked-best; and the weighted intersection between the reranked 50-best list and the SPRep system.

and Johnson (2005), which allows a comparison of the shallow parsing gains that they obtain from that system with those documented here. The reranker is a discriminatively trained Maximum Entropy model with an F-measure parsing accuracy objective. It uses a large number of features, and is applied to the 50-best output from the generative Charniak parsing model. The reranking model was trained on sections 2-21, with section 24 used as heldout. This allows us to compare its shallow parsing accuracy with other systems on the tasks that use this training setup: the Li & Roth task (testing on WSJ section 00) and the domain shift task (testing on Switchboard section 4). Table 5 shows two new trials making use of this reranking model.

The Charniak and Johnson (2005) system output (denoted *C & J* in the table) before reranking (denoted *one-best*) is identical to the Charniak (2000) results that have been reported in the other tables. After reranking (denoted *reranked*), the performance improves by roughly 0.7 percentage points for both tasks, nearly reaching the performance that we obtained with weighted intersection of the SPRep model and the n-best list before reranking. Weighted intersection between the reranked list and the shallow parser as described earlier, with a newly estimated scaling factor (α=30), provides a roughly 0.5 percentage point increase over the result obtained by the reranker. The difference between the Charniak output before and after reranking is statistically significant at $p < 0.001$, as is the difference between the reranked output and the weighted intersection, using the same test reported earlier.

3.5 Discussion

While it may be seen to be overkill to apply a context-free parser for these shallow parsing tasks, we feel that these results are very interesting for a couple of reasons. First, they go some way toward correcting the misperception that context-free parsers are less applicable in real-world scenarios than finite-state sequence models. Finite-state models are undeniably more efficient; however, it is important to have a clear idea of how much accuracy is being sacrificed to reach that efficiency. Any given application will need to examine the efficiency/accuracy trade-off with different objectives for optimality. For those willing to trade efficiency for accuracy, it is worthwhile knowing that it is possible to do much better on these tasks than what has been reported in the past.

4 Conclusion and Future Work

In summary, we have demonstrated in this paper that there is no accuracy or robustness benefit to shallow parsing with finite-state models over using high-accuracy context-free models. Even more, there is a large benefit to be had in combining the output of high-accuracy context-free parsers with the output of shallow parsers. We have demonstrated a large improvement over the previous-best reported results on several tasks, including the well-known NP-Chunking and CoNLL-2000 shallow parsing tasks.

Part of the misperception of the relative benefits of finite-state and context-free models is due to difficulty evaluating across these differing annotation styles. Mapping from context-free parser output to the shallow constituents defined in the CoNLL-2000 task depends on many construction-specific operations that have unfairly penalized context-free parsers in previous comparisons.

While the results of combining system outputs show one benefit of combining systems, as presented in this paper, they hardly exhaust the possibilities of exploiting the differences between these models. Making use of the scores for the shallow parses output by the Charniak parser is a demonstrably effective way to improve performance. Yet there are other possible features explicit in the context-free parse candidates, such as head-to-head dependencies, which might be exploited to further improve performance. We intend to explore including features from the context-free parser output in our perceptron model to improve shallow parsing accuracy.

Another possibility is to look at improving

context-free parsing accuracy. Within a multi-pass parsing strategy, the high-accuracy shallow parses that result from system combination could be used to restrict the search within yet another pass of a context-free parser. That parser could then search for the best global analysis from within just the space of parses consistent with the provided shallow parse. Also, features of the sort used in our shallow parser could be included in a reranker, such as that in Charniak and Johnson (2005), with a context-free parsing accuracy objective.

A third possibility is to optimize the definition of the shallow-parse phrase types themselves, for use in other applications. The composition of the set of phrase types put forth by Sang and Buchholz (2000) may not be optimal for certain applications. One such application is discourse parsing, which relies on accurate detection of clausal boundaries. Shallow parsing could provide reliable information on the location of these boundaries, but the current set of phrase types may be too general for such use. For example, consider infinitival verb phrases, which often indicate the start of a clause whereas other types of verb phrases do not. Unfortunately, with only one VP category in the CoNLL-2000 set of phrase types, this distinction is lost. Expanding the defined set of phrase types could benefit many applications.

Future work will also include continued exploration of possible features that can be of use for either shallow parsing models or context-free parsing models. In addition, we intend to investigate ways in which to encode approximations to context-free parser derived features that can be used within finite-state models, thus perhaps preserving finite-state efficiency while capturing at least some of the accuracy gain that was observed in this paper.

Acknowledgments

We would like to thank Eugene Charniak and Mark Johnson for help with the parser and reranker documented in their paper. The first author of this paper was supported under an NSF Graduate Research Fellowship. In addition, this research was supported in part by NSF Grant #IIS-0447214. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Steven Abney. 1996. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4).

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

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of ACL*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Annual Meeting of NAACL*, pages 132–139.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of ACL*, pages 16–23.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th ICML Conference*.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on EMNLP*, pages 1–8.

L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of ICASSP*, pages 532–535.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the 2nd Annual Meeting of NAACL*.

Xin Li and Dan Roth. 2001. Exploring evidence for shallow parsing. In *Proceedings of the 5th Conference on CoNLL*.

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.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 82–94.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on CoNLL*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the HLT-NAACL Annual Meeting*.

Tong Zhang, Fred Damerau, and David Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.