Low-Level Mark-Up and Large-scale LFG Grammar Processing

Ronald M. Kaplan and Tracy Holloway King
Palo Alto Research Center

Proceedings of the LFG03 Conference
University at Albany, State University of New York
Miriam Butt and Tracy Holloway King (Editors)

2003

CSLI Publications
http://csli-publications.stanford.edu/

Abstract

It is commonly believed that shallow mark-up techniques such as part-of-speech disambiguation or low-level phrase chunking provide useful information that can improve the performance of natural language processing systems, even those that ultimately require deeper levels of analysis. In this paper, we discuss three types of shallow mark-up: part of speech tagging, named entities, and labeled bracketing. We show how they were integrated into the ParGram LFG English grammar and report on the results of parsing the PARC700 sentences with each type of mark-up. We observed that named-entity mark-up improves both speed and accuracy and labeled brackets also can be beneficial, but that part-of-speech tags are not particularly useful.

1 Introduction

It is commonly believed that shallow mark-up techniques such as part-of-speech (POS) disambiguation or low-level phrase chunking provide useful information that can improve the performance of natural language processing systems, even those that ultimately require deeper levels of analysis.[1] This should be the case to the extent that they reduce ambiguity while still preserving the correct POS or bracketing.

These shallow mark-up techniques typically operate independently of the deeper analysis machinery. They take input strings in normal orthography and modify those strings by adding diacritic marks to record information. We discuss three types of shallow mark-up: part of speech tagging, named entities, and labeled bracketing. These are illustrated in (1)–(3) and will be discussed in detail below.

(1) Part of Speech Tagging

a. I/PRP saw/VBD her/PRP duck/VB.

b. I/PRP saw/VBD her/PRP$ duck/NN.

(2) Named Entities

a. <person>General Mills</person> bought it.

b. <company>General Mills</company> bought it.

(3) Labeled Bracketing

a. [NP-SBJ I] saw [NP-OBJ the girl with the telescope].

b. [NP-SBJ I] saw [NP-OBJ the girl] with the telescope.

1.1 Hypothesis and Experiments

In this paper, we explore the hypothesis that:

> Ground-truth shallow mark-up reduces ambiguity and increases speed without decreasing accuracy.

[1]This research has been funded in part by contract # MDA904-03-C-0404 awarded to Stuart K. Card and Peter Pirolli from the Advanced Research and Development Activity, Novel Intelligence from Massive Data program.

We would like to thank Stefan Riezler for help in conducting the experiments described here.

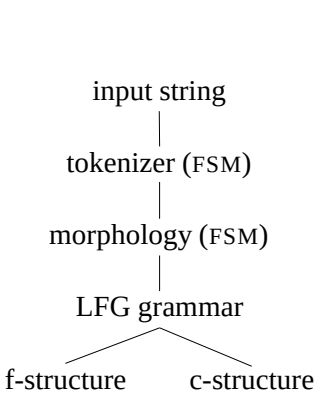
We tested this hypothesis by applying a combination of an existing broad-coverage grammar and robust parsing system to four versions of 700 sentences drawn from the hand-annotated Wall Street Journal treebank constructed at the University of Pennsylvania (Marcus et al., 1994). We compared parsing speed and accuracy for sentences with the three kinds of mark-up exemplified in (1)–(3) against a baseline of sentences without any mark-up at all. These experiments aimed at determining the best-case benefit of different kinds of mark-up in that we make use of the ground-truth annotations of the WSJ corpus, essentially simulating the fictional situation of error-free pre-processors that take no time at all to run. This provides an upper-bound on the accuracy and time benefits that could be achieved with any practical mark-up technology.

In our experiments this hypothesis is confirmed for named entity mark-up and there is also some benefit for labeled bracketing, but we observed little value for POS tagging. In the conclusion, we suggest some further refinements of the experiments to be explored in future work.

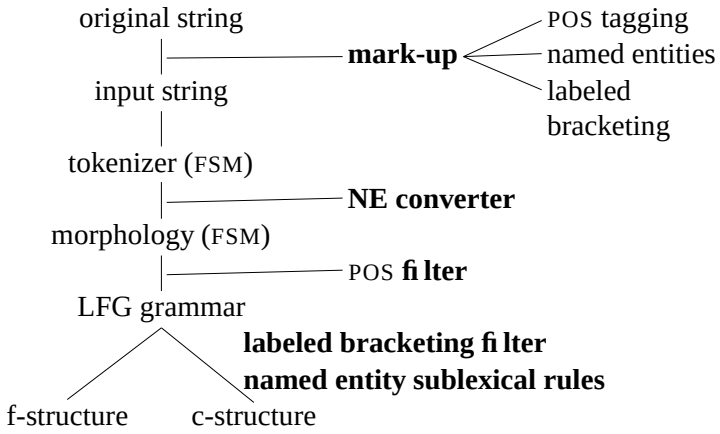
1.2 The XLE System

In this paper, we also address the problem of how to modify an existing system of morphological and syntactic analysis so that it respects the restrictions provided by independent low-level mark-up components. In our existing system (the XLE grammar development platform (Maxwell and Kaplan 1993)), tokenizing and morphological analysis are performed by finite-state transductions (Beesley and Karttunen 2003) arranged in a compositional cascade. The resulting morphological analysis is presented to an efficient parsing system for broad-coverage LFG grammars, the ParGram grammars described by Butt et al. (1999, 2002). The basic system organization is shown in (4).

(4) Original System



Integrated System



2 Integrating Low-level Mark-Up

We confront the following issue: how can a system defined to operate on ordinary sentences be extended to handle input with diacritical mark-up so that the set of syntactic analyses is limited to those consistent with the specified mark-up. This devolves into two questions:

(5) a. How does the mark-up move through the sequence of components so that it reaches the level at which its constraints are imposed?

b. How are those constraints imposed?

In this section we discuss the integration of POS tagging, named entities, and labeled bracketing into the ParGram English grammar used by the XLE system.

2.1 Part of Speech Tagging

POS tags are not relevant to tokenization, but the tokenizing transducer must be modified so that the POS tags do not interfere with the other patterns that the tokenizer is concerned with (commas, quotes, etc.).[2] The proper effect of a POS tag is to reduce the number of outputs from the morphological analyzer. Consider the words *likes* and *walks* in (6a), which have the POS mark-up in (6b) (other POS tags are not shown in (6b)).

(6) a. She likes to go on walks.

b. She likes/VBZ to go on walks/NNS .

The morphological analyzer normally would produce two outputs for each of these words (plural noun and third-person singular verb) of which only one is appropriate in this context, the verbal interpretation for *likes* and the noun interpretation for *walks*.

Obtaining the desired behavior requires a specification of the morphological output sequences that are consistent with a given POS tag. In this case, we must specify that VBZ is consistent with +Verb +Pres +3sg but not compatible with +Noun, and that NNS is consistent with +Noun +Pl but not +Verb. This is done by a mapping table that states legal correspondences between POS tags and the morphological analyzer tags. Some sample mappings between the WSJ Penn Treebank tags and our morphology tags are shown in (7). Some POS tags correspond to more than one set of morphological tags, e.g. NNS in (7).

Given this mapping table, we produce a finite-state machine that filters the normal output of the morphological transducer. The filtering machine, for example, allows pairs of lemmas and morphological indicators to be followed by VBZ if and only if the indicators are compatible with the allowable sequences drawn from the table. Thus the indicator +Noun is not allowed in front of VBZ. The filtering machine also maps the POS tags to epsilon, so that they do not appear in its output. This machine is then put in a cascade with the tokenizer and morphological analyzer. The overall effect is that the tags in the input are preserved by the tokenizer, pass through the morphological analyzer, and then cause incompatible morphological-indicator sequences to be discarded. Thus only the contextually appropriate interpretations of *likes* and *walks* survive, and these are all that the syntactic grammar has to operate on.

[2] A tokenizer splits a string into words or tokens. Minimally, this involves splitting off punction marks. In our system, tokenization also involves determining which words should be lower cased (e.g., sentence initial words are optionally lowered cased). Finally, our tokenizer deals with haplology; for example, it optionally inserts commas before sentence periods. A sample input string with some possible tokenizations is shown in (i). Note that tokenizers can produce multiple outputs for a single input string; the tokenization used by the grammar to parse the sentence in (i.a) is that in (i.b).

(i) a. Input string:
They walked the dog, a poodle.

b. Splitting, decapping, haplology:
they walked the dog , a poodle , .

c. Splitting, decapping:
they walked the dog , a poodle .

d. Splitting, haplology:
They walked the dog , a poodle , .

(7) **Mapping POS Tags to Morphological Tags**

POS tag	Morphological tag(s)	Interpretation
IN	+Prep	*preposition*
	+Conj +Subord	*subordinating conjunction*
JJR	+Adj +Comp	*comparative adjective*
NNS	+Noun +Pl	*plural noun*
	+Noun +SP	*noun that can be singular or plural (sheep)*
	+Abbr	*plural abbreviation*
	+Num +Fract +Pl	*plural fraction*
	+Meas	*measure phrase*
VBG	+Verb +PresPart	*present participle*
	+Verb +Prog	*progressive verb*
	+Aux +Prog	*progressive auxiliary*
VBN	+Verb +PastPart	*past participle*
	+Verb +PastBoth	*past participle or past tense*
VBZ	+Verb +Pres +3sg	*present third singular verb*
	+Aux +Pres +3sg	*present third singular auxiliary*
WRB	+Adv +IntRel	*interrogative or relative adverb*

2.2 Named Entities

Next consider named entities, multi-word sequences that refer to particular entities, such as people or companies.[3] The named entities appear in the text as XML mark-up, as in (8).

(8) <company>General Mills</company> bought it.

The tokenizer was modifi edto include an additional tokenization of the strings whereby the material between the XML mark-up was treated as a single token with a special morphological tag on it, as in (9a). Here the underscore represents the literal space occurring in the middle of the named-entity. As a fall back mechanism, the tokenization that the string would have received without that mark-up is also produced, as in (9b).

(9) a. General_Mills +NamedEntity bought it .

b. General Mills bought it .

The morphological analyzer was then modifi edto allow the additional morphological indicator +NamedEntity to pass through. The lexicon was extended to recognize that indicator and to provide suitable f-structure features for it (e.g., person, number, proper). The grammar was changed to allow that indicator to follow nouns. In addition, an optimality mark (Frank et al. 2001) was used to prefer the named entity reading over the other reading when possible; when a parse cannot be built with the named entity reading, then the other tokenization is tried. A skeletal f-structure for (9a) is shown in (10).

[3] Common nouns, such as dates and numbers, can also be marked up as named entities, but we did not use these in our experiments. The technique for integrating these named entities into the system would be identical to that described here for proper nouns, although the lexical entries for these tags would assign slightly different features.

(10)
$$\begin{bmatrix} \text{PRED} & \text{'buy<SUBJ, OBJ>'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'General\_Mills'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{PRED} & \text{'pro'} \end{bmatrix} \\ \text{TENSE} & \text{past} \end{bmatrix}$$

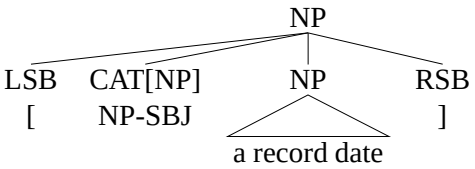
Note that the subject predicate in (10) contains the single form 'General_Mills'. This single multi-word predicate helps in parsing both because it means that no internal structure has to be built for the predicate, which improves the speed, and because predicates that would otherwise be unrecognized by the grammar can be parsed (e.g., *Cie. Financiere de Paribas*), which improves the coverage.

2.3 Labeled Bracketing

Labeled brackets are treated much like POS tags, except that they must be preserved through the morphology so that they can be interpreted as constraints on the LFG grammar. The grammar was modified so that it parses the labeled brackets as part of the c-structure. These brackets not only force the material between them to be constituents, but they also dictate which c-structure categories are possible. For example, the NP-SBJ label only allows NP constituents. This bracketing helps to eliminate attachment ambiguities. A sample labeled bracketed sentence is shown in (11), and the c-structure of the bracketed NP is shown in (12). Additional examples of labeled bracketing are shown in (13).

(11) [NP-SBJ A record date] hasn't been set.

(12)



(13) a. [NP-SBJ Lloyd's] only recently reported [NP its financialresults for 1986].

b. [NP-SBJ The hall's few computers] are used mostly [VP to send [NP messages]].

c. [NP-SBJ Moody's Investors Service Inc.] said [SBAR [NP-SBJ it] lowered [NP the debt ratings of certain long-term debt held by [NP-LGS this company]]].

In addition, the lexical entries for the labels use inside-out function application to specify the grammatical function of the constituent. For example, the entry for NP-SBJ in (14) specifiesthat the f-structure corresponding to that constituent must be a SUBJ.

(14) NP-SBJ CAT[NP] (SUBJ ↑).

The categories chosen for this experiment and their corresponding grammatical function constraints are shown in (15). These categories were chosen because they encode core grammatical functions. Note that the lexical entries for these labels will also constrain the c-structure of the constituent, as described above.

(15) **Labels Used in Labeled Bracketing**

Label	F-structure constraint	Role in clause
NP	(OBJ ↑)	*direct object*
	(OBJ-TH ↑)	*secondary object*
NP-SBJ	(SUBJ ↑)	*subject*
S-NOM-SBJ	(SUBJ ↑)	*clausal subject*
SBAR-SBJ	(SUBJ ↑)	*clausal subject*
SBAR-NOM-SBJ	(SUBJ ↑)	*clausal subject*
NP-LGS	(OBL-AG ↑)	*demoted subject of passive*
S-NOM-LGS	(OBL-AG ↑)	*demoted clausal subject of passive*
ADJP-PRD	(XCOMP ↑)	*predicative adjective*
ADV-PRD	(XCOMP ↑)	*predicative adverb*
NP-PRD	(XCOMP ↑)	*predicative nominal*
PP-PRD	(XCOMP ↑)	*predicative prepositional phrase*
S-PRD	(XCOMP ↑)	*predicative clause*
S-NOM-PRD	(XCOMP ↑)	*predicative clause*
SBAR-PRD	(XCOMP ↑)	*predicative clause*

C-structure constraint only (no f-structure constraint):
NP-EXT, NP-TMP
SBAR, SBAR-TMP, SBAR-ADV
PP
VP

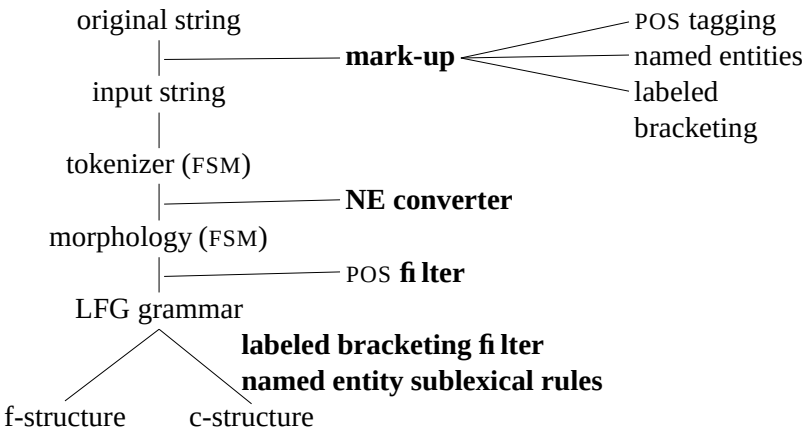
2.4 The Combined System

POS tagging, named entities, and labeled bracketing can be combined in the input, as in (16). As described above, the POS tags (VBZ, VBN) are filtered before the grammar parses the marked up input, and the named entities (<company> ... </company>) are processed then as well, while the labeled brackets ([NP-SBJ ...]) are filtered by the grammar.

(16) [NP-SBJ <company>American Cyanamid Co.</company>] has/VBZ n't been/VBN bought/VBN .

The integrated system is diagramed in (17), repeated here from (4). A preprocessing step has been added to insert the shallow mark-up into the input string, the FSTs have been altered to filter the POS tags and the named entities, and the grammar has been slightly modified to parse the labeled bracketing and the named entities tags. As with the original system, the output of the integrated system is the usual LFG c-structure and f-structure analyses of the input string.

(17) Integrated System



3 Accuracy of the Integrated System

We have shown how this kind of information can be integrated into a deep parsing system. That raises the question, is this information (POS, labeled brackets, named entities) reliably available, and if so, does it actually help? We first discuss the issue of reliability and then whether the gold standard mark-up helped with ambiguity, speed, and accuracy in our system.

3.1 Gold Standard Mark-Up

For the our initial experiment which is reported here, we used gold standard shallow mark-up. For the POS tagging this was obtained by extracting the POS tags from the UPenn WSJ treebank. Since this corpus was hand annotated, the POS tags are assumed to be correct. The gold standard for the named entities was created by the authors. To bootstrap our gold standard, we used an in-house version of Fact Finder, a finite-statenamed entity recognizer developed at PARC. Names of companies, organizations, and people were marked. The gold standard for the labeled bracketing was extracted from the UPenn WSJ treebank. As discussed above, a subset of the brackets were chosen to reflect core grammatical functions.

The reasoning behind using this gold standard mark-up was that if perfect shallow mark-up does not improve ambiguity and speed without sacrificingaccuracy, then automatically produced mark-up is bound to fail. In the paragraphs below, we briefly discuss issues with using automatically produced shallow mark-up (also see section 4.2).

Many POS taggers are available, but for English they have about 96% accuracy when run on sentences drawn from the same corpus as the taggers are trained on. An error rate of 1 tag every 20 words is probably not acceptable, since on average there would be one tagging error in every 20 word sentence. Thus the utility of this requires a tagger that guarantees 100% recall of the correct tag even if it must provide several alternative outputs in cases where it might otherwise make an error. It remains to be seen whether this is sufficient to have the desired effects on processing efficiency and on reducing ambiguity.

With syntactic brackets, there is a somewhat different problem, since there is no standard online technology that provides a syntactically useful phrase-chunker. Still, this integration is helpful for doing experiments on manually labeled treebanks, such as the Penn Tree bank (Marcus et al. 1994). However, even with these, only certain bracketings are compatible with the LFG grammar. In our initial experiment, we extracted constituents which indicated core grammatical relations from the

Penn treebank (see section 2.3).

There are a number of named entity finders available, some are stochastic while others use finite-state technology. As mentioned, to bootstrap our gold standard, we used an in-house named-entity recognizer, the Fact Finder. Fact Finder is extremely fast and, unlike most other named entity recognizers, it can produce ambiguous output, as in (18). This is ideal for a preprocessing step in a system that includes deeper levels of analysis.

(18) <person><company>General Mills</company></person>

We have not made a detailed evaluation of this version of Fact Finder or of other entity finders. However, we conducted one run of our grammar on a non-gold standard version of the Fact Finder output. The results still showed an improvement over the unmarked strings. As such, we are optimistic that for at least some domains, such as newspaper texts which contain many complex proper nouns, using an entity finder as a preprocessor will improve accuracy and speed.

3.2 Results

To test the accuracy and speed of the gold-standard shallow mark-up, we ran four test suites on the integrated grammar. The first was the unmarked strings corresponding to the sentences in the PARC700 test set (King et al. 2003). This is a subset of the sentences in Section 23 of the UPenn WSJ treebank, the standard testing section. The second test suite was comprised of these same strings with gold-standard named entity mark-up. The third was these same strings with gold-standard POS tagging extracted from the UPenn WSJ treebank. Finally, the forth was these same strings with gold-standard labeled bracketing also extracted from the UPenn WSJ treebank.

We compared the resulting f-structures with the PARC700 dependency bank to determine how accurate the results were. We separated out the result for sentences that receive full parses from those that receive FRAGMENT parses, parses that are produced by the fall-back robustness mechanisms of XLE and the grammar.[4] In general, full parses are more accurate than FRAGMENT parses, since the grammatical relations that would have connected the FRAGMENTs together are not recovered.

(19) **Results on the PARC700 (WSJ)**

	%Full parses	Solutions	Best f-score	Time %
Unmarked	76	482/1753	82/79	65/100
Named Entities	78	263/1477	86/84	60/91
POS tag	62	248/1916	76/72	40/48
Labeled bracketing	65	158/774	85/79	19/31

Note: the scores with / are full parses/all parses

The table is to be read as follows. Consider the first row which describes the parse results for the unmarked-up strings (i.e., the input to the original system).

[4]The FRAGMENT grammar allows the input to be analyzed as a sequence of well-formed chunks. These chunks are specified by the grammar, for example Ss, NPs, PPs, and VPs. These chunks have both c- and f-structures corresponding to them. Any token that cannot be parsed as one of these chunks is parsed as a TOKEN chunk. The TOKENs are also recorded in the c- and f-structures. The grammar has a fewest chunk method for determining the correct parse. For example, if a string can be parsed as two NPs and a VP or as one NP and an S, the NP-S option is chosen. For an example FRAGMENT parse from the XLE analysis of the WSJ Penn Treebank, see Riezler et al. 2002.

- When parsing the PARC700, 76% of these strings got full parses.[5] That is, they have both a spanning c-structure and a well-formed f-structure. This is a measure of *coverage* for our hand-written, corpus-independent grammar.
- The full parses had an average of 482 solutions; many of the sentences had relatively few solutions while a few had a very large number of solutions. If full and FRAGMENT parses are considered, the average number of solutions was 1753. This is a measure of *ambiguity* for the parses.
- The average best f-score for full parses was 82, while that for all of the parses was 79. The best f-score is the average of precision and recall for grammatical relations (e.g., SUBJ, OBJ) for the parse of each sentence that best matched the gold-standard. This is our measure of parse *accuracy*.
- The *time* for parsing all of the sentences was set at 100% for the unmarked strings. The full parses accounted for 65% of this total time. The time for all-parse processing is always higher because of the additional effort required by our fall-back robustness techniques.

The other rows of the table show the results for input with named entities, with POS tagging, and with labeled bracketing. We discuss these in turn.

Named Entities (% full: 78; soln: 263/1477; f-score: 86/84; time: 60/91)
The named entity mark-up was the most successful of the three types of mark-up. Full coverage increased in that the number of full parses went from 76% to 78%. In addition, ambiguity dropped so that there were only 263 average solutions for the full parses. The accuracy also increased in that the f-score for full parses went from 82 to 86 and from 79 to 84 for all parses; these are substantial gains. Finally, there was a modest improvement in speed in that parsing the entire 700 sentences took only 91% of the time that parsing the unmarked strings did. We see these benefits because expensive and erroneous analyses of the internal structure of named entities are avoided.

POS **Tagging** (% full: 62; soln: 248/1916; f-score: 76/72; time: 40/48)
The POS tagging was not as successful. In particular, the number of full parses fell significantly from 76% to only 62% and the accuracy of these parses also decreased as witnessed by the average f-score of 76 for the full parses, compared to 82 for the unmarked strings. The only improvements came from ambiguity for the full parses and from speed. The speed for the POS tagging was about half of that for the unmarked strings. In the results presented here, we used full POS tagging. That is, every word was marked for POS. In future work, we hope to explore whether using only partial tagging, such as just verbs and nouns, would work better in that it would decrease the time and the ambiguity without hurting coverage or accuracy.

Labeled Bracketing (% full: 65; soln: 158/774; f-score: 85/79; time: 19/31)
The situation with the labeled bracketing was better than that of the POS tagging. The worst result was that coverage decreased from 76% full parses to only 65% full parses. However, the ambiguity decreased to only an average of 158 solutions for full parses, compared with 482 for unmarked strings, and the average best f-score for full parses increased to 85%. Even with the decline in coverage, the f-score for all parses remained at 79, the same value as for unmarked strings, suggesting

[5] Current coverage of the grammar is over 80%; this improvement should benefit both unmarked strings and ones with shallow mark-up.

that the increased number of FRAGMENT parses still retained, or even slightly improved, in accuracy. This was accomplished with a dramatic improvement in speed in that the labeled bracketed sentences took only a third of the time of the unmarked strings. Thus, if speed is important and an overall f-score comparable to the much slower unmarked parsing is acceptable (and if a reliable and effi cientpre-processor for labeled bracketing is available), this kind of mark-up may be very useful.

4 Conclusions and Discussion

Our results show that further experiments need to be done as to the ultimate feasibility of integrating low-level mark-up into deep parsing. In particular, we saw that there was an immediate gain both in speed and accuracy from named entity mark-up, and there was a speed gain for labeled bracketing without serious degradation of accuracy. However, POS tagging, even when using gold standard mark-up, resulted in a signifi cantloss in coverage and accuracy in that the number of full parses dropped from that of unmarked strings. An interesting additional experiment would be to determine what combination of types of shallow mark-up are best; for example, we could combine named entities with POS tagging of verbs.

4.1 Fall-back Techniques

One area for improvement is fi ndingbetter fall-back techniques when the low level mark-up fails. Whenever the grammar cannot create an analysis which is compatible with the mark-up, the system uses a fall-back technique. These fall-back techniques involve a second parsing pass and hence can signifi cantlyslow the overall parsing time.

In the case of the named entities, the fall-back technique is to use the output of the tokenizer that ignores the named entity mark-up entirely; these two outputs were seen in (9). In our experience so far, this works quite well, in part because it is needed relatively rarely.

The fall-back for the labeled bracketing may also be working relatively well, as witnessed by the relatively low ambiguity rate even when all the parses are considered (158/774 vs. 482/1753 for unmarked strings). When the grammar cannot create a well-formed analysis for a labeled bracketed sentence, it uses the grammar's general robustness mechanism of building well-formed FRAGMENTs (see fn. 4 and Riezler et al. 2002). Note that each FRAGMENT may contain labeled brackets that are correctly parsed. At this point, we have not investigated the effect of the labeled bracketing on the FRAGMENT grammar in depth, and so there may be ways in which to decrease the ambiguity of the FRAGMENT parses further. In particular, unlike named entities, the labeled brackets are always parsed, even in the FRAGMENTS; that is, they are always part of the c-structure tree. The grammar cannot try an alternative where just the unmarked string is parsed. It is unclear whether being able to do this would be an advantage or not.

In contrast to the named entities and labeled bracketing, we have not found a good fall-back mechanism for the POS tags. In one attempt, we created a version of the grammar that allowed the grammar to ignore the POS tag restrictions if no parse could be found that obeyed them. Although this improved the accuracy to be roughly that of the unmarked strings (since effectively it allowed the grammar to parse the unmarked string), the time was extremely slow, slower than parsing the unmarked strings. We thus abandoned this approach. One possibility to then consider is to use partial POS tagging instead of full POS tagging in hopes that there will be a set of POS tags that improve ambiguity and time while not hurting accuracy.

4.2 On-the-fly Mark-up

If these results are favorable, the question remains whether on-the-fly mark-up, as opposed to mark-up extracted from a manually annotated treebank, can be used without erroneously eliminating correct parses. Some initial experiments with named entity mark-up indicate that automatic named entity mark-up still improves results over unmarked strings. As such, we are very optimistic about integrating named entity mark-up into the system.

In contrast, POS tagging is less likely to be successful. Its success will depend on whether there is some combination of tags, e.g., nouns and verbs, which give reasonable coverage with improved accuracy and ambiguity and on whether these tags are the types that POS taggers can assign with very good accuracy. This is particularly important because we have not yet found a good fall-back mechanism for when POS matching fails.

Finally, we are also optimistic about labeled bracketing, given the speed and accuracy we observed in these experiments. However, we know of no chunker or parser that can produce the type of input we need with enough accuracy, and the speed of that pre-processing might also be a serious component of overall cost. In other words, producing the mark-up itself is a serious issue. Thus, our current primary use for labeled bracketing is to test the accuracy of our grammar. That is, if we find a full parse for a given bracketing, we can be relatively certain that the grammar has assigned the sentence a correct f-structure.

References

- K. Beesley and L. Karttunen. 2003. *Finite-State Morphology.* CSLI Publications.
- M. Butt, H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The Parallel Grammar Project. *Proceedings of COLING2002, Workshop on Grammar Engineering and Evaluation* pp. 1-7.
- M. Butt, T.H. King, M.-E. Niño, and F. Segond. 1999. *A Grammar Writer's Cookbook.* CSLI Publications.
- A. Frank, T.H. King, J. Kuhn, and J.T. Maxwell III. 2001. Optimality Style Constraint Ranking in Large-scale LFG Grammars. In P. Sells (ed.) *Formal and Empirical Issues in Optimality Theoretic Syntax.* CSLI Publications.
- T.H. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03).*
- M. Marcus, G. Kim, M.A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Furguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating Predicate Argument Structure. In *ARPA Human Language Technology Workshop*.
- J.T. Maxwell, III and R. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Lingusitics*, 19:571–589.
- S. Riezler, T.H. King, R. Kaplan, R. Crouch, J. Maxwell, and M. Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. *Proceedings of the Annual Meeting of the Association for Computational Linguistics, University of Pennsylvania.*