Accurate Function Parsing

Paola Merlo
Department of Linguistics
University of Geneva
1204 Geneva
Switzerland
merlo@lettres.unige.ch

Gabriele Musillo
Depts of Linguistics and Computer Science
University of Geneva
1204 Geneva
Switzerland
musillo4@etu.unige.ch

Abstract

In this paper, we extend an existing parser to produce richer output annotated with function labels. We obtain state-of-the-art results both in function labelling and in parsing, by automatically relabelling the Penn Treebank trees. In particular, we obtain the best published results on semantic function labels. This suggests that current statistical parsing methods are sufficiently general to produce accurate shallow semantic annotation.

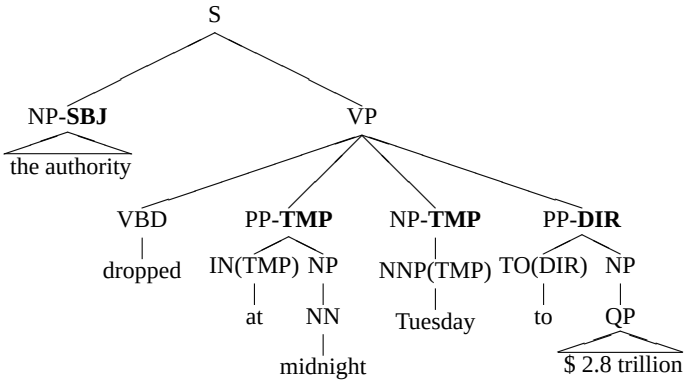

Figure 1: A sample syntactic structure with function labels.

1 Introduction

With recent advances in speech recognition, parsing, and information extraction, some domain-specific interactive systems are now of practical use for tasks such as question-answering, flight booking, or restaurant reservation (Stallard, 2000). One of the challenges ahead lies in moving from hand-crafted programs of limited scope to robust systems independent of a given domain. While this ambitious goal will remain in the future for some time to come, recent efforts to develop language processing systems producing richer semantic outputs will likely be the cornerstone of many successful developments in natural language understanding.

In this paper, we present a parser that outputs labels indicating the syntactic or semantic function of a constituent in the tree, such as NP-SBJ or PP-TMP shown in bold face in the tree in Figure 1. These labels indicate that the NP is the subject of the sentence and that the PP conveys temporal information. (Labels in parentheses will be explained later in the paper.) Output annotated with such informative labels underlies all domain-independent question answering (Jijkoun et al., 2004) or shallow semantic interpretation systems (Collins and Miller, 1998; Ge and Mooney, 2005). We test the hypothesis that a current statistical parser can output such richer information without any degradation of the parser's accuracy on the original parsing task. Briefly, our method consists in augmenting a state-of-the-art statistical parser (Henderson, 2003), whose architecture and properties make it particularly adaptive to new tasks. We achieve state-of-the-art results both for parsing and function labelling.

Statistical parsers trained on the Penn Treebank (PTB) (Marcus et al., 1993) produce trees annotated with bare phrase structure labels (Collins, 1999; Charniak, 2000). The trees of the Penn Treebank, however, are also decorated with function labels. Figure 1 shows the simplified tree representation with function labels for a sample sentence from the Penn Treebank corpus (section 00) *The Government's borrowing authority dropped at midnight Tuesday to 2.8 trillion from 2.87 trillion*. Table 1 illustrates the complete list of function labels in the Penn Treebank. Unlike phrase structure labels, func-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 620–627, Vancouver, October 2005.

Syntactic Labels		Semantic Labels	
DTV	dative	ADV	adverbial
LGS	logical subject	BNF	benefactive
PRD	predicate	DIR	direction
PUT	compl of *put*	EXT	extent
SBJ	surface subject	LOC	locative
VOC	vocative	MNR	manner
Miscellaneous Labels		NOM	nominal
CLF	*it*-cleft	PRP	purpose or reason
HLN	headline	TMP	temporal
TTL	title	**Topic Labels**	
CLR	closely related	TPC	topicalized

Table 1: Complete set of function labels in the Penn Treebank.

tion labels are context-dependent and encode a shallow level of phrasal and lexical semantics, as observed first in (Blaheta and Charniak, 2000).[1] To a large extent, they overlap with semantic role labels as defined in PropBank (Palmer et al., 2005).

Current statistical parsers do not use this richer information because performance of the parser usually decreases considerably, since a more complex task is being solved. (Klein and Manning, 2003), for instance report a reduction in parsing accuracy of an unlexicalised PCFG from 77.8% to 72.9% if using function labels in training. (Blaheta, 2004) also reports a decrease in performance when attempting to integrate his function labelling system with a full parser. Conversely, researchers interested in producing richer semantic outputs have concentrated on two-stage systems, where the semantic labelling task is performed on the output of a parser, in a pipeline architecture divided in several stages (Gildea and Jurafsky, 2002). See also the common task of (CoNLL, 2004 2005; Senseval, 2004).

Our approach maintains state-of-the-art results in parsing, while also reaching state-of-the-art results in function labelling, by suitably extending a Simple Synchrony Network (SSN) parser (Henderson, 2003) into a single integrated system. This is an interesting result, as a task combining function labelling and parsing is more complex than simple parsing. While the function of a constituent and its structural position are often correlated, they sometimes diverge. For example, some nominal temporal modifiers occupy an object position without being objects, like *Tuesday* in the tree above. Moreover, given current limited availability of annotated tree banks, this more complex task will have to be solved with the same overall amount of data, aggravating the difficulty of estimating the model's parameters due to sparse data.

[1](Blaheta and Charniak, 2000) talk of function *tags*. We will instead use the term function *label*, to indicate function identifiers, as they can decorate any node in the tree. We keep the word *tag* to indicate only those labels that decorate preterminal nodes in a tree – part-of-speech tags– as is standard use.

2 Method

Successfully addressing function parsing requires accurate parsing models and training data. Understanding the causes and the relevance of the observed results requires appropriate evaluation measures. In this section, we describe the methodology that will be used to assess our main hypothesis.

2.1 The Basic Parsing Architecture

Our main hypothesis says that function labels can be successfully and automatically recovered while parsing, without affecting negatively the performance of the parser. It is possible that attempting to solve the function labelling and the parsing problem at the same time would require modifying existing parsing models, since their underlying independence assumptions might no longer hold. Moreover, many more parameters are to be estimated. It is therefore important to choose a statistical parser that can model our augmented labelling problem. We use a family of statistical parsers, the Simple Synchrony Network (SSN) parsers (Henderson, 2003), which crucially do not make any explicit independence assumptions, and learn to smooth across rare feature combinations. They are therefore likely to adapt without much modification to the current problem. This architecture has shown state-of-the-art performance and is very adaptive to properties of the input.

The architecture of an SSN parser comprises two components, one which estimates the parameters of a stochastic model for syntactic trees, and one which searches for the most probable syntactic tree given the parameter estimates. As with many other statistical parsers (Collins, 1999; Charniak, 2000), the model of parsing is history-based. Its events are derivation moves. The set of well-formed sequences of derivation moves in this parser is defined

by a Predictive LR pushdown automaton (Nederhof, 1994), which implements a form of left-corner parsing strategy.[2]

The probability of a phrase-structure tree is equated to the probability of a finite (but unbounded) sequence of derivation moves. To bound the number of parameters, standard history-based models partition the set of prefixes of well-formed sequences of transitions into equivalence classes. While such a partition makes the problem of searching for the most probable parse polynomial, it introduces hard independence assumptions: a derivation move only depends on the equivalence class to which its history belongs. SSN parsers, on the other hand, do not state any explicit independence assumptions: they induce a finite history representation of an unbounded sequence of moves, so that the representation of a move $i-1$ is included in the inputs to the representation of the next move i, as explained in more detail in (Henderson, 2003). SSN parsers only impose soft inductive biases to capture relevant properties of the derivation, thereby exhibiting adaptivity to the input. The art of designing SSN parsers consists in selecting and introducing such biases. To this end, it is sufficient to specify features that extract some information relevant to the next derivation move from previous ones, or some set of nodes that are structurally local to the node on top of the stack. These features and these nodes are input to the computation of a hidden history representation of the sequence of previous derivation moves. Given the hidden representation of a derivation, a log-linear distribution over possible next moves is computed. Thus, the set D of structurally local nodes and the set f of predefined features determine the inductive bias of an SSN system. Unless stated otherwise, for each of the experiments reported here, the set D that is input to the computation of the history representation of the derivation moves $d_1, \ldots, d_{i-1}$ includes the following nodes: top_i, the node on top of the pushdown stack before the ith move; the left-corner ancestor of top_i; the leftmost child of top_i; and the most recent child of top_i, if any. The set of features f includes the last move in the derivation, the label or tag of top_i, the tag-word pair of the most recently shifted word, the leftmost tag-word pair that top_i dominates.

[2]The derivation moves include: projecting a constituent with a specified label, attaching one constituent to another, and shifting a tag-word pair onto the pushdown stack.

2.2 The Set of Function Labels

The bracketting guidelines for the Penn Treebank II list 20 function labels, shown in Table 1 (Bies et al., 1995). Based on their description in the Penn Treebank guidelines, we partition the set of function labels into four classes, as indicated in the table. Following (Blaheta and Charniak, 2000), we refer to the first class as syntactic function labels, and to the second class as semantic function labels. In the rest of the paper, we will ignore the other two classes, for they do not intersect with PropBank labels, and they do not form natural classes. Like previous work (Blaheta and Charniak, 2000), we complete the sets of syntactic and semantic labels by labelling constituents that do not bear any function label with a NULL label.[3]

2.3 Evaluation

To evaluate the performance of our function parsing experiments, we will use several measures. First of all, we apply the standard Parseval measures of labelled recall and precision to a parser whose training data contains the Penn Treebank function labels, to assess how well we solve the standard phrase structure parsing problem. We call these figures FLABEL-less figures in the tables below and we will call the task the (simple) parsing task in the rest of the paper. Second, we measure the accuracy of this parser with an extension of the Parseval measures of labelled precision and recall applied to the set of complex labels —the phrase structure non-terminals augmented with function labels— to evaluate how well the parser solves this complex parsing problem. These are the FLABEL figures in the tables below. We call this task the function parsing task. Finally, we also assess function labelling performance on its own. Note that the maximal precision or recall score of function labelling is strictly smaller than one-hundred percent if the precision or the recall of

[3]Strictly speaking, this label corresponds to two NULL labels: SYN-NULL and SEM-NULL. A node bearing the SYN-NULL label is a node that does not bear any other syntactic label. Analogously, the SEM-NULL label completes the set of semantic labels. Note that both the SYN-NULL label and the SEM-NULL are necessary, since both a syntactic and a semantic label can label a given constituent.

		ASSIGNED LABELS										
		ADV	BNF	DIR	EXT	LOC	MNR	NOM	PRP	TMP	SEM-NULL	SUM
ACTUAL LABELS	ADV	143	0	0	0	0	0	0	1	3	11	158
	BNF	0	0	0	0	0	0	0	0	0	1	1
	DIR	0	0	39	0	3	4	0	0	1	51	98
	EXT	0	0	0	37	0	0	0	0	0	17	54
	LOC	0	0	1	0	345	3	0	0	15	148	512
	MNR	0	0	0	0	3	35	0	0	16	40	94
	NOM	2	0	0	0	0	0	88	0	0	4	94
	PRP	0	0	0	0	0	0	0	54	1	33	88
	TMP	18	0	1	0	24	11	0	1	479	105	639
	SEM-NULL	12	0	13	5	81	28	12	24	97	20292	20564
	SUM	175	0	54	42	456	81	100	80	612	20702	22302

Table 2: Confusion matrix for simple baseline model, tested on the validation set (section 24 of PTB).

the parser is less than one-hundred percent. Following (Blaheta and Charniak, 2000), incorrectly parsed constituents will be ignored (roughly 11% of the total) in the evaluation of the precision and recall of the function labels, but not in the evaluation of the parser. Of the correctly parsed constituents, some bear function labels, but the overwhelming majority do not bear any label, or rather, in our notation, they bear a NULL label. To avoid calculating excessively optimistic scores, constituents bearing the NULL label are not taken into consideration for computing overall recall and precision figures. NULL-labelled constituents are only needed to calculate the precision and recall of other function labels. (In other words, NULL-labelled constituents never contribute to the numerators of our calculations.) For example, consider the confusion matrix M in Table 2, which reports scores for the semantic labels recovered by the baseline model described below. Precision is computed as $\frac{\sum_{i\in\{\text{ADV}\cdots\text{TMP}\}} M[i,i]}{\sum_{j\in\{\text{ADV}\cdots\text{TMP}\}} M[\text{SUM},j]}$. Recall is computed analogously. Notice that $M[n,n]$, that is the [SEM-NULL,SEM-NULL] cell in the matrix, is never taken into account.

3 Learning Function Labels

In order to assess the complexity of the task of predicting function labels while parsing, we run first the SSN on the function parsing task, without modifications to the parser. The confusion matrix for semantic function labels of this simple baseline model is illustrated in Table 2.

It is apparent that the baseline model's largest cause of error is confusion between the labels and the NULL label. These misclassfications affect recall in particular. Consider, for example, the MNR label, where 40 out of 94 occurrences are not given a function label. We add two augmentations to the parser to alleviate this problem.

The simple baseline parser treats NULL labels like other labels, and it does not distinguish subtypes of NULL labels. Our first augmentation of the parser is designed to discriminate among constituents with these NULL labels. We hypothesize that the label NULL (ie. SYN-NULL and SEM-NULL) is a mixture of types, which will be more accurately learnt separately. If the label NULL is learnt more precisely, the recall of the other labels will increase. The NULL label in the training set was automatically split into the mutually exclusive labels CLR, OBJ and OTHER. Constituents were assigned the OBJ label according to the conditions stated in (Collins, 1999).[4]

Another striking property of the simple baseline function parser is that the SSN tends to project NULL labels more than any other label. Since SSNs decide the label of a non-terminal at projection, this behaviour indicates that the parser does not have enough information at this point in the parse to project the correct function label. We hypothesize that finer-grained labelling will improve parsing performance. This observation is consistent with results reported in (Klein and Manning, 2003), who showed that part-of-speech tags occurring in the Treebank are not fine-grained enough to discriminate between

[4]Roughly, an OBJ non-terminal is an NP, SBAR or S whose parent is an S, VP or SBAR. Any such non-terminal must not bear either syntactic or semantic function labels, or the CLR label. In addition, the first child following the head of a PP is marked with the OBJ label.

preterminals. For example, the tag TO labels both the preposition *to* and the infinitival marker. Extending (Klein and Manning, 2003)'s technique to function labelling, we split some part-of-speech tags into tags marked with semantic function labels. More precisely, we concentrate on the function labels DIR, LOC, MNR, PRP or TMP, which appear to cause the most trouble to the parser, as illustrated in Table 2.

The label attached to a non-terminal was propagated down to the pre-terminal tag of its head. The labels in parentheses in Figure 1 illustrate the effect of this lowering of the labels. The goal of this tag-splitting is to indicate more clearly to the parser what kind of label to project on reading a word-tag pair in the input. To this end, re-labelling is applied only if the non-terminal dominates the pre-terminal immediately. This constraint guarantees that only those non-terminals that are actual projections of the pre-terminal are affected by this tag-splitting method. Linguistically, we are trying to capture the notion of maximal projection. [5] This augmented model has a total of 188 non-terminals to represent labels of constituents, instead of the 33 of the original SSN parser. As a result of lowering the five function labels, 83 new part-of-speech tags were introduced to partition the original tagset of the Treebank. There are 819 tag-word pairs in this model, while the original SSN parser has a vocabulary size of 508 tag-word pairs. These augmented tags as well as the 155 new non-terminals are included in the set f of features input to parsing decisions as described in section 2.1.

SSN parsers do not tag their input sentences. To provide the augmented model with tagged input sentences, we trained an SVM tagger whose features and parameters are described in detail in (Gimenez and Marquez, 2004). Trained on section 2-21, the tagger reaches a performance of 95.8% on the test set (section 23) of the PTB using our new tag set.

4 Experiments

In this section, we report the results of the experiments testing hypotheses concerning our function parser. All SSN function parsers were trained on sections 2-21 from the Penn Treebank, validated on section 24, and tested on section 23. All models are trained on parse trees whose labels include function labels. Both results taking function labels into account (FLABEL) and results not taking them into account (FLABEL-less) are reported. All our models, as well as the parser described in (Henderson, 2003), are run only once. [6] These results are reported in Table 3.

	FLABEL			FLABEL-less		
	F	R	P	F	R	P
			Validation Set			
Base	83.4	82.8	83.9	87.7	87.1	88.2
Aug	84.6	84.0	85.2	88.1	87.5	88.7
			Test Set			
Aug	86.1	85.8	86.5	88.9	88.6	89.3
H03				88.6	88.3	88.9

Table 3: Percentage F-measure (F), recall (R), and precision (P) of the SSN baseline (Base) and augmented (Aug) parsers. H03 indicates the model illustrated in (Henderson, 2003).

Our hypothesis states that we can perform function labelling and parsing at the same time, without loss in parsing performance. For this to be an interesting statement, we need to show that function labelling is not a straightforward extension of simple parsing. If simple parsing could be easily applied to function parsing, we should not have a degradation of an SSN parser model evaluated on the complex labels, compared to the same SSN parser evaluated only on phrase structure labels. As the results on the validation set indicate, our baseline model with function labels (FLABEL) is indeed lower than the performance of the parser when function labels are not taken into account (FLABEL-less), indicating that the function parsing task is more difficult than the simple parsing task.

Since the function parsing problem is more difficult than simple parsing, it is then interesting to observe that performance of the augmented parser increases significantly (FLABEL column) ($p < .001$) without losing accuracy on the parsing task

[5] This condition was relaxed in a few cases to capture constructs such as coordinated PPs (e.g. $[_{\text{PP-LOC}}[_{\text{PP}}[_{\text{IN}}at]\ldots][_{\text{CC}}and][_{\text{PP}}[_{\text{IN}}in]\ldots]\ldots]$ or infinitival clauses (e.g. $[_{\text{S-PRP}}[_{\text{VP}}[_{\text{TO}}to][_{\text{VP}}[_{\text{VB}}\ldots]\ldots]\ldots]$).

[6] This explains the little difference in performance between our results for H03 and those cited in (Henderson, 2003), where the best of three runs on the validation set is chosen.

(FLABEL-less column), compared to the initial parsing performance (as indicated by the performance of H03). Notice that, numerically, we do in fact a little better than H03, but this difference is not significant.[7]

Beside confirming that learning function labels does not increase parsing errors, we can also confirm that the nature of the errors remains the same. A separate comparison of labelled and unlabelled scores of our complex function parser indicates that unlabelled results are roughly 1% better than labelled results (F measure 89.8% on the validation set). The original SSN parser exhibits the same differential. This shows that, like other simple parsers, the function parser makes mostly node attachment mistakes rather than labelling mistakes.

A separate experiment only discriminating NULL labels indicates that this modification is indeed useful, but not as much as introducing new tags, on which we concentrate to explain the results. There is converging evidence indicating that the improvement in performance is due to having introduced new tag-word pairs, and not simply new words. First of all, of the 311 new tag-word pairs only 122 introduce truly new words. The remaining pairs are constituted by words that were already in the original vocabulary and have been retagged, or by tags associated to unknown words.

Second, this interpretation of the results is confirmed by comparing different ways of enlarging the vocabulary size input to the SSN. (Henderson, 2003) tested the effect of larger input vocabulary on SSN performance by changing the frequency cut-off that selects the input tag-word pairs. A frequency cut-off of 200 yields a vocabulary of 508 pairs, while a cut-off of 20 yields 4242 pairs, 3734 of which comprise new words. This difference in input size does not give rise to an appreciable difference in performance. On the contrary, we observe that introducing 122 new words and 83 new tags improves results considerably. This leads us to conclude that the performance of the augmented model is not simply due to a larger vocabulary.

We think that our tag-word pairs are effective because they are selected by a linguistically meaningful criterion and are more informative exemplars for the parser. Instead, simply decreasing the frequency cut-off adds mostly types of words for which the parser already possesses enough evidence (in general, nouns). Our method of lowering function labels acts as a finer-grained classification that partitions different kinds of complements based on their lexical semantic characteristics, yielding classes that are relevant to constituent structure. For instance, it is well known that lexical semantic properties of arguments of verbs are related to the verb's argument structure, and consequently to the parse tree that the verb occurs in. Partitioning a verb's complements into function classes could influence attachment decisions beneficially. We also think that the parser we use is particularly able to take advantage of these subclasses. One of the main properties of SSN parsers is that they do not need large vocabularies, because the SSN is good at generalising item-specific properties into an internal hidden representation of word classes.

Finally, to provide a meaningful and complete evaluation of the parser, it is necessary to examine the level of performance on the function labels for those constituents that are correctly parsed according to the usual Parseval measure, i.e. for those constituents for which the phrase structure labels and the string covered by the label have been correctly

	Syntactic Labels			Semantic Labels		
	F	R	P	F	R	P
	Validation Set					
Base	95.3	93.9	96.7	73.1	70.2	76.3
Aug	95.7	95.0	96.5	80.1	77.0	83.5
	Test Set					
Aug	96.4	95.3	97.4	86.3	82.4	90.5
BC00	95.7	95.8	95.5	79.0	77.6	80.4
B04 FT	95.9	95.3	96.4	83.4	80.3	86.7
B04 KP	98.7	98.4	99.0	78.0	73.2	83.5

Table 4: Percentage F-measure (F), recall (R), and precision (P) function labelling, separated for syntactic and semantic labels, for our models and Blaheta and Charniak's (BC00) and Blaheta's models (B04 FT, B04 KP). The feature trees (FT) and kernel perceptrons (KP) are optimised separately for the two different sets of labels.

[7] Significance was measured with the randomized significance test described in (Yeh, 2000).

	Baseline		Augmented	
	P	R	P	R
ADV	81.7	90.5	87.9	81.0
DIR	72.2	39.8	77.0	48.5
EXT	88.1	68.5	86.8	63.5
LOC	75.7	67.4	78.9	74.6
MNR	43.2	37.2	74.0	55.7
NOM	88.0	93.6	88.7	93.1
PRP	67.5	61.4	74.4	65.9
TMP	78.3	75.0	89.6	83.7

Table 5: Percentage F-measure (F), recall (R), and precision (P) function labelling, separated for individual semantic labels, for validation set.

recovered. Clearly, our parsing results would be uninteresting if our recall on function labels were very low. In that case, we would have failed to learn the function parsing task, and that would trivially yield a good performance on the simple parsing task. Table 4 reports the aggregated numbers for the baseline and the augmented model, while Table 5 reports separate figures for each semantic function label. These tables show that we also perform well on the labelling task alone. [8] Comparison to other researchers (last three lines of Table 4) shows that we achieve state-of-the-art results with a single integrated model that is jointly optimised for all the different types of function labels and for parsing, while previous attempts are optimised separately for the two different sets of labels. In particular, our method performs better on semantic labels.

5 Related Work

As far as we are aware, there is no directly comparable work, as nobody has so far attempted to fully merge function labelling or semantic role labelling into parsing. We will therefore discuss separately those pieces of work that have made limited use of function labels for parsing (Klein and Manning, 2003), and those that have concentrated on recovering function labels as a separate task (Blaheta and Charniak, 2000; Blaheta, 2004). We cannot discuss here the large recent literature on semantic role labelling for reasons of space, apart from work that also recovers function labels (Jijkoun and de Rijke, 2004) and work that trains a parser on Propbank labels as the first stage of a semantic role labelling pipeline (Yi and Palmer, 2005).

(Klein and Manning, 2003) and, to a much more limited extent, (Collins, 1999) are the only researchers we are aware of who used function labels for parsing. In both cases, the aim was actually to improve parser performance, consequently only few carefully chosen labels were used. (Klein and Manning, 2003) suggest the technique of tag splitting for the constituent bearing the label TMP. They also speculate that locative labels could be fruitfully percolated down the tree onto the preterminals. Results in Table 5 indicate more precisely that lowering locative labels does indeed bring about some improvement, but not as much as the MNR and TMP labels.

In work that predates the availability of Framenet and Propbank, (Blaheta and Charniak, 2000) define the task of function labelling for the first time and highlight its relevance for NLP. Their method is in two-steps. First, they parse the Penn Treebank using a state-of-the-art parser (Charniak, 2000). Then, they assign function labels using features from the local context, mostly limited to two levels up the tree and only one next label. (Blaheta, 2004) extends on this method by developing specialised feature sets for the different subproblems of function labelling and slightly improves the results, as reported in Table 4. (Jijkoun and de Rijke, 2004) approach the problem of enriching the output of a parser in several steps. The first step applies memory-based learning to the output of a parser mapped to dependency structures. This step learns function labels. Only aggregated results for all function labels, and not only for syntactic or semantic labels, are provided. Although they cannot be compared directly to our results, it is interesting to notice that they are slightly better in F-measure than Blaheta's (F=88.5%). (Yi and Palmer, 2005) share the motivation of our work, although they apply it to a different task. Like the current work, they observe that the distributions of semantic labels could potentially interact with the distributions of syntactic labels and redefine the boundaries of constituents, thus yielding trees that reflect generalisations over both these sources of information.

[8] See also (Musillo and Merlo, 2005) for more detail and comparisons on the labelling task.

6 Conclusions

In this paper we have presented a technique to extend an existing parser to produce richer output, annotated with function labels. We show that both state-of-the-art results in function labelling and in parsing can be achieved. Application of these results are many-fold, such as information extraction or question answering where shallow semantic annotation is necessary. The technique illustrated in this paper is of wide applicability to all other semantic annotation schemes available today, such as Propbank and Framenet, and can be easily extended. Work to extend this technique to Propbank annotation is underway. Since function labels describe dependence relations between the predicative head and its complements, whether they be arguments or adjuncts, this paper suggests that a left-corner parser and its probabilistic model, which are defined entirely on configurational criteria, can be used to produce a dependency output. Consequences of this observation will be explored in future work.

Acknowledgments

We thank the Swiss National Science Foundation for its support of this research under grant number 105286. We thank James Henderson for allowing us to use his parser and for numerous helfpul discussions.

References

Ann Bies, M. Ferguson, K.Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style. Technical report, University of Pennsylvania.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Procs of NAACL'00*, pages 234–240, Seattle, Washington.

Don Blaheta. 2004. *Function Tagging*. Ph.D. thesis, Department of Computer Science, Brown University.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Procs of NAACL'00*, pages 132–139, Seattle, Washington.

Michael Collins and Scott Miller. 1998. Semantic tagging using a probabilistic context-free grammar. In *Procs of the Sixth Workshop on Very Large Corpora*, pages 38–48, Montreal, CA.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, University of Pennsylvania.

CoNLL. 2004, 2005. Conference on computational natural language learning (conll-2004/05).

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Procs of CONLL-05*, Ann Arbor, Michigan.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jesus Gimenez and Lluis Marquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Procs of LREC'04*, Lisbon, Portugal.

Jamie Henderson. 2003. Inducing history representations for broad-coverage statistical parsing. In *Procs of NAACL-HLT'03*, pages 103–110, Edmonton, Canada.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Procs of ACL'04*, pages 311–318, Barcelona,Spain.

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Procs of COLING-2004*, Geneva, Switzerland.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Procs of ACL'03*, pages 423–430, Sapporo, Japan.

Mitch Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Gabriele Musillo and Paola Merlo. 2005. Assigning function labels to unparsed text. In *Procs of RANLP'05*, Korovets, Bulgaria.

Mark Jan Nederhof. 1994. *Linguistic Parsing and Program Transformations*. Ph.D. thesis, Department of Computer Science, University of Nijmegen.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.

Senseval. 2004. Third international workshop on the evaluation of systems for the semantic analysis of text (acl 2004). http://www.senseval.org/senseval3.

David Stallard. 2000. Talk'n'travel: A conversational system for air travel planning. In *Procs of ANLP'00*, pages 68–75, Seattle, Washington.

Alexander Yeh. 2000. More accurate tests for the statistical significance of the result differences. In *Procs of COLING 2000*, pages 947–953, Saarbrucken, Germany.

Szu-ting Yi and Martha Palmer. 2005. The integration of semantic parsing and semantic role labelling. In *Procs of CoNLL'05*, Ann Arbor, Michigan.