Lexical and Structural Biases for Function Parsing

Gabriele Musillo
Depts of Linguistics and Computer Science
University of Geneva
2 Rue de Candolle
1211 Geneva 4
Switzerland
musillo4@etu.unige.ch

Paola Merlo
Department of Linguistics
University of Geneva
2 Rue de Candolle
1211 Geneva 4
Switzerland
merlo@lettres.unige.ch

Abstract

In this paper, we explore two extensions to an existing statistical parsing model to produce richer parse trees, annotated with function labels. We achieve significant improvements in parsing by modelling directly the specific nature of function labels, as both expressions of the lexical semantics properties of a constituent and as syntactic elements whose distribution is subject to structural locality constraints. We also reach state-of-the-art accuracy on function labelling. Our results suggest that current statistical parsing methods are sufficiently robust to produce accurate shallow functional or semantic annotation, if appropriately biased.

1 Introduction

Natural language processing methods producing shallow semantic output are starting to emerge as the next step towards successful developments in natural language understanding. Incremental, robust parsing systems will be the core enabling technology for interactive, speech-based question answering and dialogue systems. In recent years, corpora annotated with semantic and function labels have seen the light (Palmer et al., 2005; Baker et al., 1998) and semantic role labelling has taken centre-stage as a challenging new task. State-of-the-art statistical parsers have not yet responded to this challenge.

State-of-the-art 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, labels that indicate the grammatical and semantic relationship of phrases to each other in the sentence. Figure 1 shows the simplified tree representation with function labels for a sample sentence from the PTB corpus (section 00) *The Government's borrowing authority dropped at midnight Tuesday to 2.80 trillion from 2.87 trillion*. Unlike phrase structure labels, function labels are context-dependent and encode a shallow level of phrasal and lexical semantics, as observed first in (Blaheta and Charniak, 2000). For example, while *the authority* in Figure 1 will always be a Noun Phrase, it could be a subject, as in the example, or an object, as in the sentence *They questioned his authority*, depending on its position in the sentence. To some extent, function labels overlap with semantic role labels as defined in PropBank (Palmer et al., 2005). Table 1

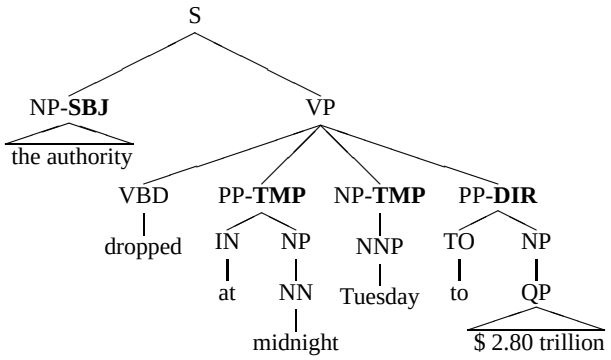

Figure 1: A sample syntactic structure with function labels.

Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT), pages 83–92,
Vancouver, October 2005. ©2005 Association for Computational Linguistics

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.

illustrates the complete list of function labels in the Penn Treebank, partitioned into four classes. [1]

Current statistical parsers do not use or output 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; Nielsen and Pradhan, 2004; Xue and Palmer, 2004). See also the common task of (CoNLL, 2004; CoNLL, 2005; Senseval, 2004), where parsing has sometimes not been used and has been replaced by chunking.

In this paper, we present a parser that produces richer output using information available in a corpus incrementally. Specifically, the parser outputs additional labels indicating the function of a constituent in the tree, such as NP-SBJ or PP-TMP in the tree shown in Figure 1.

Following (Blaheta and Charniak, 2000), we concentrate on syntactic and semantic function labels. We will ignore the other two classes, for they do not form natural classes. Like previous work, constituents that do not bear any function label will receive a NULL label. Strictly speaking, this label corresponds to two NULL labels: the SYN-NULL and the 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.

We present work to test the hypothesis that a current statistical parser (Henderson, 2003) can output richer information robustly, that is without any degradation of the parser's accuracy on the original parsing task, by explicitly modelling function labels as the locus where the lexical semantics of the elements in the sentence and syntactic locality domains interact. Briefly, our method consists in augmenting the parser with features and biases that capture both lexical semantics projections and structural regularities underlying the distribution of sequences of function labels in a sentence. We achieve state-of-the-art results both in parsing and function labelling. This result has several consequences.

On the one hand, we show that it is possible to build a single integrated robust system successfully. This is an interesting achievement, 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. Solving this more complex problem successfully, then, indicates that the models used are robust. Our results also provide some new insights into the discussion about the necessity of parsing for function or semantic role labelling (Gildea and Palmer, 2002; Punyakanok et al.,

[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.

2005), showing that parsing is beneficial.

On the other hand, function labelling while parsing opens the way to interactive applications that are not possible in a two-stage architecture. Because the parser produces richer output incrementally at the same time as parsing, it can be integrated in speech-based applications, as well as be used for language models. Conversely, output annotated with more informative labels, such as function or semantic labels, underlies all domain-independent question answering (Jijkoun et al., 2004) or shallow semantic interpretation systems (Collins and Miller, 1998; Ge and Mooney, 2005).

2 The Basic Architecture

To achieve the complex task of assigning function labels while parsing, we use a family of statistical parsers, the Simple Synchrony Network (SSN) parsers (Henderson, 2003), which do not make any explicit independence assumptions, and are therefore likely to adapt without much modification to the current problem. This architecture has shown state-of-the-art performance.

SSN parsers comprise 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), SSN parsers use a history-based model of parsing. Events in such a model 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.

This pushdown automaton operates on configurations of the form (Γ, v), where Γ represents the stack, whose right-most element is the top, and v the remaining input. The initial configuration is $(ROOT, w)$ where $ROOT$ is a distinguished non-terminal symbol. The final configuration is $(ROOT, \epsilon)$. Assuming standard notation for context-free grammars (Nederhof, 1994), three derivation moves are defined:

shift

$([B \rightarrow \beta], av) \vdash ([B \rightarrow \beta][A \rightarrow a], v)$
where $A \rightarrow a$ and $B \rightarrow \beta C \gamma$ are productions such that A is a left-corner of C.

project

$([B \rightarrow \beta][A \rightarrow \alpha], v) \vdash$
$([B \rightarrow \beta][D \rightarrow A], v)$
where $A \rightarrow \alpha$, $D \rightarrow A\delta$ and $B \rightarrow \beta C \gamma$ are productions such that D is a left-corner of C.

attach

$([B \rightarrow \beta][A \rightarrow \alpha], v) \vdash ([B \rightarrow \beta A], v)$
where both $A \rightarrow \alpha$ and $B \rightarrow \beta A \gamma$ are productions.

The joint probability of a phrase-structure tree and its terminal yield can be 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 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 use a neural network architecture, called Simple Synchrony Network (Henderson and Lane, 1998), to induce a finite history representation of an unbounded sequence of moves. The history representation of a parse history $d_1, \ldots, d_{i-1}$, which we denote $h(d_1, \ldots, d_{i-1})$, is assigned to the constituent that is on the top of the stack before the ith move.

The representation $h(d_1, \ldots, d_{i-1})$ is computed from a set f of features of the derivation move d_{i-1} and from a finite set D of recent history representations $h(d_1, \ldots, d_j)$, where $j < i - 1$. Because the history representation computed for the move $i-1$ is included in the inputs to the computation of the representation for the next move i, virtually any information about the derivation history could flow from history representation to history representation and be used to estimate the probability of a derivation move. However, the recency preference exhibited by recursively defined neural networks biases learning towards information which flows through fewer

history representations. (Henderson, 2003) exploits this bias by directly inputting information which is considered relevant at a given step to the history representation of the constituent on the top of the stack before that step. To determine which history representations are input to which others and provide SSNs with a linguistically appropriate inductive bias, the set D includes history representations which are assigned to constituents that are structurally local to a given node on the top of the stack. In addition to history representations, the inputs to $h(d_1, \ldots, d_{i-1})$ include hand-crafted features of the derivation history that are meant to be relevant to the move to be chosen at step i. 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 most recent history representation of the following nodes: top_i, the node on top of the pushdown stack before the ith move; the left-corner ancestor of top_i (that is, the second top-most node on the parser's stack); 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, and the leftmost tag-word pair that top_i dominates. Given the hidden history representation $h(d_1, \cdots, d_{i-1})$ of a derivation, a normalized exponential output function is computed by SSNs to estimate a probability distribution over the possible next derivation moves d_i.[2]

The second component of SSN parsers, which searches for the best derivation given the parameter estimates, implements a severe pruning strategy. Such pruning handles the high computational cost of computing probability estimates with SSNs, and renders the search tractable. The space of possible derivations is pruned in two different ways. The first pruning occurs immediately after a tag-word pair has been pushed onto the stack: only a fixed beam of the 100 best derivations ending in that tag-word pair are expanded. For training, the width of such beam is set to five. A second reduction of the search space prunes the space of possible project or attach derivation moves: the best-first search strategy is applied to the five best alternative decisions only.

[2]The on-line version of Backpropagation is used to train SSN parsing models. It performs the gradient descent with a maximum likelihood objective function and weight decay regularization (Bishop, 1995).

3 Learning Lexical Projection and Locality Domains of Function Labels

Recent approaches to functional or semantic labels are based on two-stage architectures. The first stage selects the elements to be labelled, while the second determines the labels to be assigned to the selected elements. While some of these models are based on full parse trees (Gildea and Jurafsky, 2002; Blaheta, 2004), other methods have been proposed that eschew the need for a full parse (CoNLL, 2004; CoNLL, 2005). Because of the way the problem has been formulated, – as a pipeline of parsing feeding into labelling – specific investigations of the interaction of lexical projections with the relevant structural parsing notions during function labelling has not been studied.

The starting point of our augmentation of SSN models is the observation that the distribution of function labels can be better characterised structurally than sequentially. Function labels, similarly to semantic roles, represent the interface between lexical semantics and syntax. Because they are projections of the lexical semantics of the elements in the sentence, they are projected bottom-up, they tend to appear low in the tree and they are infrequently found on the higher levels of the parse tree, where projections of grammatical, as opposed to lexical, elements usually reside. Because they are the interface level with syntax, function and semantic labels are also subject to distributional constraints that govern syntactic dependencies, especially those governing the distribution of sequences of long distance elements. These relations often correspond to top-down constraints. For example, languages like Italian allow inversion of the subject (the Agent) in transitive sentences, giving rise to a linear sequence where the Theme precedes the Agent (*Mangia la mela Gianni, eats the apple Gianni*). Despite this freedom in the linear order, however, it is never the case that the *structural* positions can be switched. It is a well-attested typological generalisation that one does not find sentences where the subject is a Theme and the object is the Agent. The hierarchical description, then, captures the underlying generalisa-

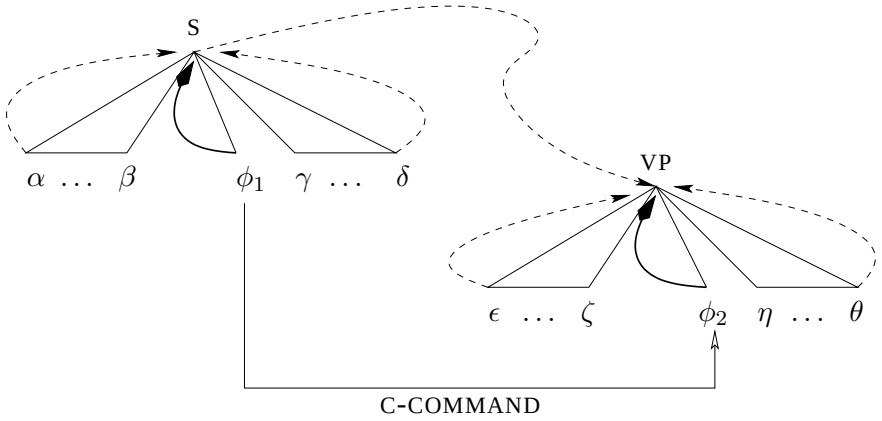

Figure 2: Flow of information in an SSN parser (dashed lines), enhanced by biases specific to function labels to capture the notion of c-command (solid lines).

tion better than a model based on a linear sequence.

In our augmented model, inputs to each history representation are selected according to a linguistically motivated notion of structural locality over which dependencies such as argument structure or subcategorization could be specified. We attempt to capture the sequence and the structural position by indirectly modelling the main definition of syntactic domain, the notion of c-command. Recall that the c-command relation defines the domain of interaction between two nodes in a tree, even if they are not close to each other, provided that the first node dominating one node also dominates the other. This notion of c-command captures both linear and hierarchical constraints and defines the domain in which semantic role labelling applies, as well as many other linguistic operations.

In SSN parsing models, the set D of nodes that are structurally local to a given node on the top of the stack defines the structural distance between this given node and other nodes in the tree. Such a notion of distance determines the number of history representations through which information passes to flow from the representation assigned to a node i to the representation assigned to a node j. By adding nodes to the set D, one can shorten the structural distance between two nodes and enlarge the locality domain over which dependencies can be specified. To capture a locality domain appropriate for function parsing, we include two additional nodes in the set D: the most recent child of top_i labelled with a syntactic function label and the most recent child of top_i labelled with a semantic function label. These additions yield a model that is sensitive to regularities in structurally defined sequences of nodes bearing function labels, within and across constituents. First, in a sequence of nodes bearing function labels within the same constituent – possibly interspersed with nodes not bearing function labels – the structural distance between a node bearing a function label and any of its right siblings is shortened and constant. This effect comes about because the representation of a node bearing a function label is directly input to the representation of its parent, until a farther node with a function label is attached. Second, the distance between a node labelled with a function label and any node that it c-commands is kept constant: since the structural distance between a node $[A \rightarrow \alpha]$ on top of the stack and its left-corner ancestor $[B \rightarrow \beta]$ is constant, the distance between the most recent child node of B labelled with a function label and any child of A is kept constant. This modification of the biases is illustrated in Figure 2.

This figure displays two constituents, S and VP with some of their respective child nodes. The VP node is assumed to be on the top of the parser's stack, and the S one is supposed to be its left-corner ancestor. The directed arcs represent the information that flows from one node to another. According to the original SSN model in (Henderson, 2003), only the information carried over by the leftmost child and the most recent child of a constituent di-

rectly flows to that constituent. In the figure above, only the information conveyed by the nodes α and δ is directly input to the node S. Similarly, the only bottom-up information directly input to the VP node is conveyed by the child nodes ϵ and θ. In both the no-biases and H03 models, nodes bearing a function label such as ϕ_1 and ϕ_2 are not directly input to their respective parents. In our extended model, information conveyed by ϕ_1 and ϕ_2 directly flows to their respective parents. So the distance between the nodes ϕ_1 and ϕ_2, which stand in a c-command relation, is shortened and kept constant.

As well as being subject to locality constraints, functional labels are projected by the lexical semantics of the words in the sentence. We introduce this bottom-up lexical information by fine-grained modelling of function tags in two ways. On the one hand, extending a technique presented in (Klein and Manning, 2003), we split some part-of-speech tags into tags marked with semantic function labels. The labels attached to a non-terminal which appeared to cause the most trouble to the parser in a separate experiment (DIR, LOC, MNR, PRP or TMP) were propagated down to the pre-terminal tag of its head. To affect only labels that are projections of lexical semantics properties, the propagation takes into account the distance of the projection from the lexical head to the label, and distances greater than two are not included. Figure 3 illustrates the result of the tag splitting operation.

On the other hand, we also split the NULL label into mutually exclusive labels. We hypothesize that the label NULL (ie. SYN-NULL and SEM-NULL) is a mixture of types, some of which of semantic nature, such as CLR, which will be more accurately learnt separately. The NULL label was split into the mutually exclusive labels CLR, OBJ and OTHER. Constituents were assigned the OBJ label according to the conditions stated in (Collins, 1999). 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. (For more detail on this lexical semantics projection, see (Merlo and Musillo, 2005).)

We report the effects of these augmentations on parsing results in the experiments described below.

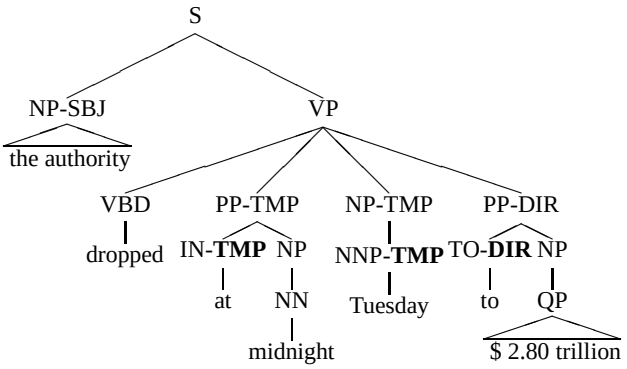

Figure 3: A sample syntactic structure with function labels lowered onto the preterminals.

4 Experiments and Discussion

To assess the relevance of our fine-grained tags and history representations for functional labelling, we compare two augmented models to two baseline models without these augmentations indicated in Table 2 as no-biases and H03. The baseline called H03 refers to our runs of the parser described in (Henderson, 2003), which is not trained on input annotated with function labels. Comparison to this model gives us an external reference to whether function labelling improves parsing. The baseline called no-biases refers to a model without any structural or lexical biases, but trained on input annotated with function labels. This comparison will tell us if the biases are useful or if the reported improvements could have been obtained without explicit manipulation of the parsing biases.

All SSN function parsers were trained on sections 2-21 from the PTB and validated on section 24. They are trained on parse trees whose labels include syntactic and semantic function labels. The models, as well as the parser described in (Henderson, 2003), are run only once. This explains the little difference in performance between our results for H03 in our table of results and those cited in (Henderson, 2003), where the best of three runs on the validation set is chosen. To evaluate the performance of our function parsing experiments, we extend standard Parseval measures of labelled recall and precision to include function labels.

The augmented models have a total of 188 non-terminals to represents labels of constituents, instead of the 33 of the baseline H03 parser. As a result

	FLABEL			FLABEL-less		
	F	R	P	F	R	P
H03				88.6	88.3	88.9
no-biases	84.6	84.4	84.9	88.2	88.0	88.4
split-tags	86.1	85.8	86.5	88.9	88.6	89.3
split-tags+locality	86.4	86.1	86.8	89.2	88.9	89.5

Table 2: Percentage F-measure (F), recall (R), and precision (P) of the SSN baseline and augmented parsers.

of lowering the five function labels, 83 new part-of-speech tags were introduced to partition the original tag set. SSN parsers do not tag their input sentences. To provide the augmented models 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.

Both parsing results taking function labels into account in the evaluation (FLABEL) and results not taking them into account in the evaluation (FLABEL-less) are reported in Table 2, which shows results on the test set, section 23 of the PTB. Both the model augmented only with lexical information (through tag splitting) and the one augmented both with finer-grained tags and representations of syntactic locality perform better than our comparison baseline H03, but only the latter is significantly better ($p < .01$, using (Yeh, 2000)'s randomised test). This indicates that while information projected from the lexical items is very important, only a combination of lexical semantics information and careful modelling of syntactic domains provides a significant improvement.

Parsing results outputting function labels (FLABEL columns) reported in Table 2 indicate that parsing function labels is more difficult than parsing bare phrase-structure labels (compare the FLABEL column to the FLABEL-less column). They also show that our model including finer-grained tags and locality biases performs better than the one including only finer-grained tags when outputting function labels. This suggests that our model with both lexical and structural biases performs better than our no-biases comparison baseline precisely because it is able to learn to parse function labels more accurately. Comparisons to the baseline without biases indicates clearly that the observed improvements, both on function parsing and on parsing without taking function labels into consideration would not have been obtained without explicit biases.

Individual performance on syntactic and semantic function labelling compare favourably to previous attempts (Blaheta, 2004; Blaheta and Charniak, 2000). 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 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. For example, consider the confusion matrix M in Table 3 below, which reports scores for the semantic labels recovered by the no-biases model. 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.

Syntactic labels are recovered with very high accuracy (F 96.5%, R 95.5% and P 97.5%) by the model with both lexical and structural biases, and so are semantic labels, which are considerably more difficult (F 85.6%, R 81.5% and P 90.2%). (Blaheta, 2004) uses specialised models for the two types

		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 3: Confusion matrix for the no-biases baseline model, tested on the validation set (section 24 of PTB).

of function labels, reaching an F-measure of 98.7% for syntactic labels and 83.4% for semantic labels as best accuracy measure. Previous work that uses, like us, a single model for both types of labels reaches an F measure of 95.7% for syntactic labels and 79.0% for semantic labels (Blaheta and Charniak, 2000).

Although functional information is explicitly annotated in the PTB, it has not yet been exploited by any state-of-the-art statistical parser with the notable exception of the second parsing model of (Collins, 1999). Collins's second model uses a few function labels to discriminate between arguments and adjuncts, and includes parameters to generate subcategorisation frames. Subcategorisation frames are modelled as multisets of arguments that are sisters of a lexicalised head child. Some major differences distinguish Collins's subcategorisation parameters from our structural biases. First, lexicalised head children are not explicitly represented in our model. Second, we do not discriminate between arguments and adjuncts: we only encode the distinctions between syntactic function labels and semantic ones. As shown in (Merlo, 2003; Merlo and Esteve-Ferrer, 2004) this difference does not correspond to the difference between arguments and adjuncts. Finally, our model does not implement any distinction between right and left subcategorisation frames. In Collins's model, the left and right subcategorisation frames are conditionally independent and arguments occupying a complement position (to the right of the head) are independent of arguments occurring in a specifier position (to the left of the head). In our model, no such independence assumptions are stated, because the model is biased towards phrases related to each other by the c-command relation. Such relation could involve both elements at the left and at the right of the head. Relations of functional assignments between subjects and objects, for example, could be captured.

The most important observation, however, is that modelling function labels as the interface between syntax and semantics yields a significant improvement on parsing performance, as can be verified in the FLABEL-less column of Table 2. This is a crucial observation in the light of the current approaches to function or semantic role labelling and its relation to parsing. An improvement in parsing performance by better modelling of function labels indicates that this complex problem is better solved as a single integrated task and that current two-step architectures might be missing on successful ways to improve both the parsing and the labelling task.

In particular, recent models of semantic role labelling separate input indicators of the correlation between the structural position in the tree and the semantic label, such as *path*, from those indicators that encode constraints on the sequence, such as the previously assigned role (Kwon et al., 2004). In this way, they can never encode directly the constraining power of a certain role in a given structural position onto a following node in its structural position. In our augmented model, we attempt to capture these constraints by directly modelling syntactic domains.

Our results confirm the findings in (Palmer et al., 2005). They take a critical look at some commonly used features in the semantic role labelling task, such as the *path* feature. They suggest that the path feature is not very effective because it is sparse. Its

sparseness is due to the occurrence of intermediate nodes that are not relevant for the syntactic relations between an argument and its predicate. Our model of domains is less noisy, because it can focus only on c-commanding nodes bearing function labels, thus abstracting away from those nodes that smear the pertinent relations.

(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.

Our results also confirm the importance of lexical information, the lesson drawn from (Thompson et al., 2004), who find that correctly modelling sequence information is not sufficient. Lexical information is very important, as it reflects the lexical semantics of the constituents. Both factors, syntactic domains and lexical information, are needed to significantly improve parsing.

5 Conclusions

In this paper, we have explored a new way to improve parsing results in a current statistical parser while at the same time enriching its output. We achieve significant improvements in parsing and function labelling by modelling directly the specific nature of function labels, as both expressions of the lexical semantics properties of a constituent and as syntactic elements whose distribution is subject to structural locality constraints. Differently from other approaches, the method we adopt integrates function labelling directly in the parsing process. Future work will lie in exploring new ways of capturing syntactic domains, different from the ones attempted in the current paper, such as developing new derivation moves for nodes bearing function labels. A more detailed analysis of the parser will also shed light on its behaviour on sequences of function labels. Finally, we plan to extend this work to learn Propbank-style semantic role labels, which might require explicit modelling of long distance dependencies and syntactic movement.

Acknowledgements

We thank the Swiss National Science Foundation for supporting this research under grant number 101411-105286/1. We also thank James Henderson for allowing us to use his parser and James Henderson and Mirella Lapata for useful discussion of this work. All remaining errors are our own.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics (ACL-COLING'98)*, pages 86–90, Montreal, Canada. Morgan Kaufmann Publishers.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Meeting of North American Chapter of Association for Computational Linguistics (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 *Proceedings of the 1st Meeting of North American Chapter of Association for Computational Linguistics (NAACL'00)*, pages 132–139, Seattle, Washington.

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

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

CoNLL. 2004. Eighth conference on computational natural language learning (conll-2004). http://cnts.uia.ac.be/conll2004.

CoNLL. 2005. Ninth conference on computational natural language learning (conll-2005). http://cnts.uia.ac.be/conll2005.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL-05)*, Ann Arbor, Michigan.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 239–246, Philadelphia, PA.

Jesus Gimenez and Lluis Marquez. 2004. Svmtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

James Henderson and Peter Lane. 1998. A connectionist architecture for learning to parse. In *Proceedings of 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98)*, pages 531–537, University of Montreal, Canada.

Jamie Henderson. 2003. Inducing history representations for broad-coverage statistical parsing. In *Proceedings of the Joint Meeting of the North American Chapter of the Association for Computational Linguistics and the Human Language Technology Conference (NAACL-HLT'03)*, pages 103–110, Edmonton, Canada.

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

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the ACL (ACL'03)*, pages 423–430, Sapporo, Japan.

Namhee Kwon, Michael Fleischman, and Eduard Hovy. 2004. Senseval automatic labeling of semantic roles using maximum entropy models. In *Senseval-3*, pages 129–132, Barcelona, Spain.

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

Paola Merlo and Eva Esteve-Ferrer. 2004. PP attachment and the notion of argument. University of Geneva, manuscript.

Paola Merlo and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, Canada.

Paola Merlo. 2003. Generalised PP-attachment disambiguation using corpus-based linguistic diagnostics. In *Proceedings of the Tenth Conference of The European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 251–258, Budapest, Hungary.

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

Rodney Nielsen and Sameer Pradhan. 2004. Mixing weak learners in semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 80–87, Barcelona, Spain, July.

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

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'05)*.

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

Cynthia Thompson, Siddharth Patwardhan, and Carolin Arnold. 2004. Generative models for semantic role labeling. In *Senseval-3*, Barcelona, Spain.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 88–94, Barcelona, Spain.

Alexander Yeh. 2000. More accurate tests for the statistical signifi cance of the result differences. In *Procs of the 17th International Conf. on Computatioal Linguistics*, pages 947–953, Saarbrucken, Germany.

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