Parsing with generative models of predicate-argument structure

Julia Hockenmaier
IRCS, University of Pennsylvania, Philadelphia, USA
and
Informatics, University of Edinburgh, Edinburgh, UK
juliahr@linc.cis.upenn.edu

Abstract

The model used by the CCG parser of Hockenmaier and Steedman (2002b) would fail to capture the correct bilexical dependencies in a language with freer word order, such as Dutch. This paper argues that probabilistic parsers should therefore model the dependencies in the predicate-argument structure, as in the model of Clark et al. (2002), and defines a generative model for CCG derivations that captures these dependencies, including bounded and unbounded long-range dependencies.

1 Introduction

State-of-the-art statistical parsers for Penn Treebank-style phrase-structure grammars (Collins, 1999), (Charniak, 2000), but also for Categorial Grammar (Hockenmaier and Steedman, 2002b), include models of bilexical dependencies defined in terms of local trees. However, this paper demonstrates that such models would be inadequate for languages with freer word order. We use the example of Dutch ditransitives, but our argument equally applies to other languages such as Czech (see Collins et al. (1999)). We argue that this problem can be avoided if instead the bilexical dependencies in the predicate-argument structure are captured, and propose a generative model for these dependencies.

The focus of this paper is on models for Combinatory Categorial Grammar (CCG, Steedman (2000)). Due to CCG's transparent syntax-semantics interface, the parser has direct and immediate access to the predicate-argument structure, which includes not only local, but also long-range dependencies arising through coordination, extraction and control. These dependencies can be captured by our model in a sound manner, and our experimental results for English demonstrate that their inclusion improves parsing performance. However, since the predicate-argument structure itself depends only to a degree on the grammar formalism, it is likely that parsers that are based on other grammar formalisms could equally benefit from such a model. The conditional model used by the CCG parser of Clark et al. (2002) also captures dependencies in the predicate-argument structure; however, their model is inconsistent.

First, we review the dependency model proposed by Hockenmaier and Steedman (2002b). We then use the example of Dutch ditransitives to demonstrate its inadequacy for languages with a freer word order. This leads us to define a new generative model of CCG derivations, which captures word-word dependencies in the underlying predicate-argument structure. We show how this model can capture long-range dependencies, and deal with the presence of multiple dependencies that arise through the presence of long-range dependencies. In our current implementation, the probabilities of derivations are computed during parsing, and we discuss the difficulties of integrating the model into a probabilistic chart parsing regime. Since there is no CCG treebank for other languages available, experimental results are presented for English, using CCGbank

(Hockenmaier and Steedman, 2002a), a translation of the Penn Treebank to CCG. These results demonstrate that this model benefits greatly from the inclusion of long-range dependencies.

2 A model of surface dependencies

Hockenmaier and Steedman (2002b) define a surface dependency model (henceforth: SD) **HWDep** which captures word-word dependencies that are defined in terms of the derivation tree itself. It assumes that binary trees (with parent category P) have one head child (with category H) and one non-head child (with category D), and that each node has one lexical head $h{=}\langle c, w\rangle$. In the following tree, $P{=}\mathsf{S[dcl]\backslash NP}$, $H{=}\mathsf{(S[dcl]\backslash NP)/NP}$, $D{=}\mathsf{NP}$, $h_H = \langle \mathsf{(S[dcl]\backslash NP)/NP}, \textit{opened}\rangle$, and $h_D{=}\langle \mathsf{N}, \textit{doors}\rangle$.

$$\mathsf{S[dcl]\backslash NP} \rightarrow \underset{\textit{opened}}{\mathsf{(S[dcl]\backslash NP)/NP}} \quad \underset{\textit{its doors}}{\mathsf{NP}}$$

The model conditions w_D on its own lexical category c_D, on $h_H = \langle c_H, w_H\rangle$ and on the local tree τ in which the D is generated (represented in terms of the categories $\langle P, H, D\rangle$):

$$P(w_D | c_D, \tau = \langle P, H, D\rangle, h_H = \langle c_H, w_H\rangle)$$

3 Predicate-argument structure in CCG

Like Clark et al. (2002), we define predicate-argument structure for CCG in terms of the dependencies that hold between words with lexical functor categories and their arguments. We assume that a lexical head is a pair $\langle c, w\rangle$, consisting of a word w and its lexical category c. Each constituent has at least one lexical head (more if it is a coordinate construction). The arguments of functor categories are numbered from 1 to n, starting at the innermost argument, where n is the arity of the functor, eg. $\mathsf{(S[dcl]\backslash NP_1)/NP_2}$, $\mathsf{(NP\backslash NP_1)/(S[dcl]/NP)_2}$. Dependencies hold between lexical heads whose category is a functor category and the lexical heads of their arguments. Such dependencies can be expressed as 3-tuples $\langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle$, where c is a functor category with arity $\geq i$, and $\langle c', w'\rangle$ is a lexical head of the ith argument of c.

The predicate-argument structure that corresponds to a derivation contains not only local, but also long-range dependencies that are projected from the lexicon or through some rules such as the coordination of functor categories. For details, see Hockenmaier (2003).

4 Word-word dependencies in Dutch

Dutch has a much freer word order than English. The analyses given in Steedman (2000) assume that this can be accounted for by an extended use of composition. As indicated by the indices (which are only included to improve readability), in the following examples, *hij* is the subject ($\mathsf{NP_3}$) of *geeft*, *de politieman* the indirect object ($\mathsf{NP_2}$), and *een bloem* the direct object ($\mathsf{NP_1}$).[1]

Hij (*He*	*geeft* *gives*	*de politieman* *the policeman*	*een bloem* *a flower*)
$\mathsf{S/(S/NP_3)}$	$\mathsf{((S/NP_1)/NP_2)/NP_3}$	$\mathsf{T\backslash(T/NP_2)}$	$\mathsf{T\backslash(T/NP_1)}$
		$\mathsf{T\backslash((T/NP_1)/NP_2)}$ $<\mathbf{B}$	
	$\mathsf{S/NP_3}$ $<\mathbf{B}_\times$		
S $>$			

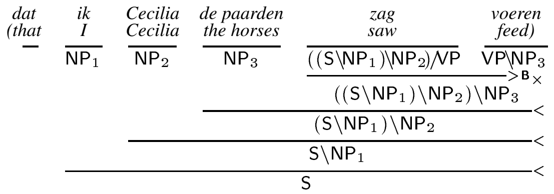
Een bloem	*geeft*	*hij*	*de politieman*
$\mathsf{S/(S/NP_1)}$	$\mathsf{((S/NP_1)/NP_2)/NP_3}$	$\mathsf{T\backslash(T/NP_3)}$	$\mathsf{T\backslash(T/NP_2)}$
	$\mathsf{(S/NP_1)/NP_2}$ $<$		
	$\mathsf{S/NP_1}$ $<$		
S $>$			

De politieman	*geeft*	*hij*	*een bloem*
$\mathsf{S/(S/NP_2)}$	$\mathsf{((S/NP_1)/NP_2)/NP_3}$	$\mathsf{T\backslash(T/NP_3)}$	$\mathsf{T\backslash(T/NP_1)}$
	$\mathsf{(S/NP_1)/NP_2}$ $<$		
	$\mathsf{S/NP_2}$ $<\mathbf{B}_\times$		
S $>$			

A SD model estimated from a corpus containing these three sentences would not be able to capture the correct dependencies. Unless we assume that the above indices are given as a feature on the NP categories, the model could not distinguish between the dependency relations of *Hij* and *geeft* in the first sentence, *bloem* and *geeft* in the second sentence and *politieman* and *geeft* in the third sentence. Even with the indices, either the dependency between *politieman* and *geeft* or between *bloem* and *geeft* in the first sentence could not be captured by a model that assumes that each local tree has exactly one head. Furthermore, if one of these sentences occurred in the training data, all of the dependencies in the other variants of this sentence would be unseen to the model. However, in terms of the predicate-argument structure, all three examples express the same relations. The model we propose here would therefore be able to generalize from one example to the word-word dependencies in the other examples.

[1]The variables T are uninstantiated for reasons of space.

The cross-serial dependencies of Dutch are one of the syntactic constructions that led people to believe that more than context-free power is required for natural language analysis. Here is an example together with the CCG derivation from Steedman (2000):



Again, a local dependency model would systematically model the wrong dependencies in this case, since it would assume that all noun phrases are arguments of the same verb.

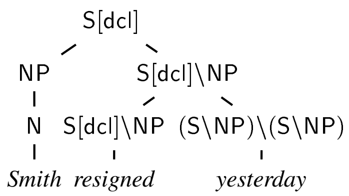
However, since there is no Dutch corpus that is annotated with CCG derivations, we restrict our attention to English in the remainder of this paper.

5 A model of predicate-argument structure

We first explain how word-word dependencies in the predicate-argument structure can be captured in a generative model, and then describe how these probabilities are estimated in the current implementation.

5.1 Modelling local dependencies

We first define the probabilities for purely local dependencies without coordination. By excluding non-local dependencies and coordination, at most one dependency relation holds for each word. Consider the following sentence:



This derivation expresses the following dependencies:

$$\langle\langle \mathsf{S[dcl]\backslash NP}, \textit{resigned}\rangle, 1, \langle \mathsf{N}, \textit{Smith}\rangle\rangle$$
$$\langle\langle \mathsf{(S\backslash NP)\backslash(S\backslash NP)}, \textit{yesterday}\rangle, 2, \langle \mathsf{S[dcl]\backslash NP}, \textit{resigned}\rangle\rangle$$

We assume again that heads are generated before their modifiers or arguments, and that word-word dependencies are expressed by conditioning modifiers or arguments on heads. Therefore, the head words of arguments (such as *Smith*) are generated in the following manner:

$$P(w_a|c_a, \langle\langle c_h,w_h\rangle, i, \langle c_a, w_a\rangle\rangle)$$

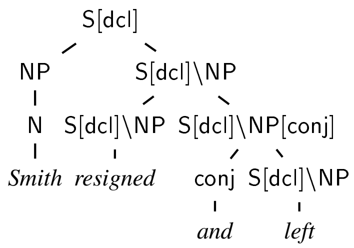
The head word of modifiers (such as *yesterday*) are generated differently:

$$P(w_m|c_m, \langle\langle c_m,w_m\rangle, i, \langle c_h, w_h\rangle\rangle)$$

Like Collins (1999) and Charniak (2000), the SD model assumes that word-word dependencies can be defined at the maximal projection of a constituent. However, as the Dutch examples show, the argument slot i can only be determined if the head constituent is fully expanded. For instance, if $\mathsf{S[dcl]}$ expands to a non-head $\mathsf{S/(S/NP)}$ and to a head $\mathsf{S[dcl]/NP}$, it is necessary to know how the $\mathsf{S[dcl]/NP}$ expands to determine which argument is filled by the non-head, even if we already know that the lexical category of the head word of $\mathsf{S[dcl]/NP}$ is a ditransitive $\mathsf{((S[dcl]/NP)/NP)/NP}$. Therefore, we assume that the non-head child of a node is only expanded after the head child has been fully expanded.

5.2 Modelling long-range dependencies

The predicate-argument structure that corresponds to a derivation contains not only local, but also long-range dependencies that are projected from the lexicon or through some rules such as the coordination of functor categories. In the following derivation, *Smith* is the subject of *resigned* and of *left*:

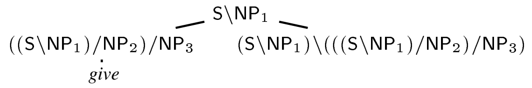


In order to express both dependencies, *Smith* has to be conditioned on *resigned* and on *left*:

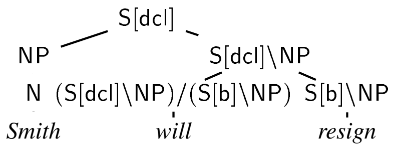
$$P(w = \textit{Smith}|\ \mathsf{N}, \langle\langle \mathsf{S[dcl]\backslash NP}, \textit{resigned}\rangle, 1, \langle \mathsf{N}, w\rangle\rangle, \langle\langle \mathsf{S[dcl]\backslash NP}, \textit{left}\rangle, 1, \langle \mathsf{N}, w\rangle\rangle)$$

In terms of the predicate-argument structure, *resigned* and *left* are both lexical heads of this sentence. Since neither fills an argument slot of the other, we assume that they are generated independently. This is different from the SD model, which conditions the head word of the second and subsequent conjuncts on the head word of the first conjunct. Similarly, in a sentence such as *Miller and Smith resigned*, the current model assumes that the two heads of the subject noun phrase are conditioned on the verb, but not on each other.

Argument-cluster coordination constructions such as *give a dog a bone and a policeman a flower* are another example where the dependencies in the predicate-argument structure cannot be expressed at the level of the local trees that combine the individual arguments. Instead, these dependencies are projected down through the category of the argument cluster:

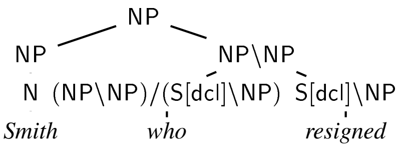


Lexical categories that project long-range dependencies include cases such as relative pronouns, control verbs, auxiliaries, modals and raising verbs. This can be expressed by co-indexing their arguments, eg. $(\mathsf{NP\backslash NP_i})/(\mathsf{S[dcl]\backslash NP_i})$ for relative pronouns. Here, *Smith* is also the subject of *resign*:

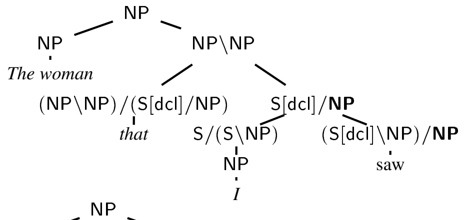


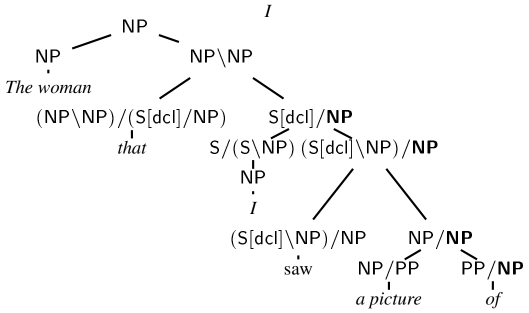
Again, in order to capture this dependency, we assume that the entire verb phrase is generated before the subject.

In relative clauses, there is a dependency between the verbs in the relative clause and the head of the noun phrase that is modified by the relative clause:



Since the entire relative clause is an adjunct, it is generated after the noun phrase *Smith*. Therefore, we cannot capture the dependency between *Smith* and *resigned* by conditioning *Smith* on *resigned*. Instead, *resigned* needs to be conditioned on the fact that its subject is *Smith*. This is similar to the way in which head words of adjuncts such as *yesterday* are generated. In addition to this dependency, we also assume that there is a dependency between *who* and *resigned*. It follows that if we want to capture unbounded long-range dependencies such as object extraction, words cannot be generated at the maximal projection of constituents anymore. Consider the following examples:





In both cases, there is a $\mathsf{S[dcl]/NP}$ with lexical head $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$; however, in the second case, the NP argument is not the object of the transitive verb.

This problem can be solved by generating words at the leaf nodes instead of at the maximal projection of constituents. After expanding the $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$ node to $(\mathsf{S[dcl]\backslash NP})/\mathsf{NP}$ and $\mathsf{NP/NP}$, the NP that is co-indexed with *woman* cannot be unified with the object of *saw* anymore.

These examples have shown that two changes to the generative process are necessary if word-word dependencies in the predicate-argument structure are to be captured. First, head constituents have to be fully expanded before non-head constituents are generated. Second, words have to be generated at the leaves of the tree, not at the maximal projection of constituents.

5.3 The word probabilities

Not all words have functor categories or fill argument slots of other functors. For instance, punctuation marks, conjunctions, and the heads of entire sentences are not conditioned on any other words. Therefore, they are only conditioned on their lexical categories. Therefore, this model contains the following three kinds of word probabilities:

1. Argument probabilities:
   $P(w|c,\langle\langle c', w'\rangle, i, \langle c, w\rangle\rangle)$
   The probability of generating word w, given that its lexical category is c and that $\langle c, w\rangle$ is head of the ith argument of $\langle c', w'\rangle$.

2. Functor probabilities:
   $P(w|c, \langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle)$
   The probability of generating word w, given that its lexical category is c and that $\langle c', w'\rangle$ is head of the ith argument of $\langle c, w\rangle$.

3. Other word probabilities: $P(w|c)$
   If a word does not fill any dependency relation, it is only conditioned on its lexical category.

5.4 The structural probabilities

Like the SD model, we assume an underlying process which generates CCG derivation trees starting from the root node. Each node in a derivation tree has a category, a list of lexical heads and a (possibly empty) list of dependency relations to be filled by its lexical heads. As discussed in the previous section, head words cannot in general be generated at the maximal projection if unbounded long-range dependencies are to be captured. This is not the case for lexical categories. We therefore assume that a node's lexical head category is generated at its maximal projection, whereas head words are generated at the leaf nodes. Since lexical categories are generated at the maximal projection, our model has the same structural probabilities as the **LexCat** model of Hockenmaier and Steedman (2002b).

5.5 Estimating word probabilities

This model generates words in three different ways—as arguments of functors that are already generated, as functors which have already one (or more) arguments instantiated, or independent of the surrounding context. The last case is simple, as this probability can be estimated directly, by counting the number of times c is the lexical category of w in the training corpus, and dividing this by the number of times c occurs as a lexical category in the training corpus:

$$\hat{P}(w|c) = \frac{C(w,c)}{C(c)}$$

In order to estimate the probability of an argument w, we count the number of times it occurs with lexical category c and is the ith argument of the lexical functor $\langle c', w'\rangle$ in question, divided by the number of times the ith argument of $\langle c', w'\rangle$ is instantiated with a constituent whose lexical head category is c:

$$\hat{P}(w|c, \langle\langle c', w'\rangle, i, \langle c, w\rangle\rangle) = \frac{C(\langle\langle c', w'\rangle, i, \langle c, w\rangle\rangle)}{\sum_{w''} C(\langle\langle c', w'\rangle, i, \langle c, w''\rangle\rangle)}$$

The probability of a functor w, given that its ith argument is instantiated by a constituent whose lexical head is $\langle c', w'\rangle$ can be estimated in a similar manner:

$$\hat{P}(w|c, \langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle) = \frac{C(\langle\langle c, w\rangle, i, \langle c', w'\rangle\rangle)}{\sum_{w''} C(\langle\langle c, w''\rangle, i, \langle c', w'\rangle\rangle)}$$

Here we count the number of times the ith argument of $\langle c, w\rangle$ is instantiated with $\langle c', w'\rangle$, and divide this by the number of times that $\langle c', w'\rangle$ is the ith argument of any lexical head with category c. For instance, in order to compute the probability of *yesterday* modifying *resigned* as in the previous section, we count the number of times the transitive verb *resigned* was modified by the adverb *yesterday* and divide this by the number of times *resigned* was modified by any adverb of the same category.

We have seen that functor probabilities are not only necessary for adjuncts, but also for certain types of long-range dependencies such as the relation between the noun modified by a relative clause and the verb in the relative clause. In the case of zero or reduced relative clauses, some of these dependencies are also captured by the SD model. However, in that model, only counts from the same type of construction could be used, whereas in our model, the functor probability for a verb in a zero or reduced relative clause can be estimated from all occurrences of the head noun. In particular, all instances of the noun and verb occurring together in the training data (with the same predicate-argument relation between them, but not necessarily with the same surface configuration) are taken into account by the new model.

To obtain the model probabilities, the relative frequency estimates of the functor and argument probabilities are both interpolated with the word probabilities $\hat{P}(w|c)$.

5.6 Conditioning events on multiple heads

In the presence of long-range dependencies and coordination, the new model requires the conditioning of certain events on multiple heads. Since it is unlikely that such probabilities can be estimated directly from data, they have to be approximated in some manner.

If we assume that all dependencies dep_i that hold for a word are equally likely, we can approximate $P(w|c, dep_1, ..., dep_n)$ as the average of the individual dependency probabilities:

$$P(w|c, dep_1, ..., dep_n) \approx \frac{1}{n}\sum_{i=1}^{n} P(w|c, dep_i)$$

This approximation is has the advantage that it is easy to compute, but might not give a good estimate, since it averages over all individual distributions.

6 Dynamic programming and beam search

This section describes how this model is integrated into a CKY chart parser. Dynamic programming and effective beam search strategies are essential to guarantee efficient parsing in the face of the high ambiguity of wide-coverage grammars. Both use the inside probability of constituents. In lexicalized models where each constituent has exactly one lexical head, and where this lexical head can only depend on the lexical head of one other constituent, the inside probability of a constituent is the probability that a node with the label and lexical head of this constituent expands to the tree below this node. The probability of generating a node with this label and lexical head is given by the outside probability of the constituent.

In the model defined here, the lexical head of a constituent can depend on more than one other word. As explained in section 5.2, there are instances where the categorial functor is conditioned on its arguments – the example given above showed that verbs in relative clauses are conditioned on the lexical head of the noun which is modified by the relative clause. Therefore, the inside probability of a constituent cannot include the probability of any lexical head whose argument slots are not all filled.

This means that the equivalence relation defined by the probability model needs to take into account not only the head of the constituent itself, but also all other lexical heads within this constituent which have at least one unfilled argument slot. As a consequence, dynamic programming becomes less effective. There is a related problem for the beam search: in our model, the inside probabilities of constituents within the same cell cannot be directly compared anymore. Instead, the number of unfilled lexical heads needs to be taken into account. If a lexical head $\langle c, w\rangle$ is unfilled, the evaluation of the probability of w is delayed. This creates a problem for the beam search strategy.

The fact that constituents can have more than one lexical head causes similar problems for dynamic programming and the beam search.

In order to be able to parse efficiently with our model, we use the following approximations for dynamic programming and the beam search: Two constituents with the same span and the same category are considered equivalent if they delay the evaluation of the probabilities of the same words and if they have the same number of lexical heads, and if the first two elements of their lists of lexical heads are identical (the same words and lexical categories). This is only an approximation to true equivalence, since we do not check the entire list of lexical heads. Furthermore, if a cell contains more than 100 constituents, we iteratively narrow the beam (by halving it in size)[2] until the beam search has no further effect or the cell contains less than 100 constituents. This is a very aggressive strategy, and it is likely to adversely affect parsing accuracy. However, more lenient strategies were found to require too much space for the chart to be held in memory. A better way of dealing with the space requirements of our model would be to implement a packed shared parse forest, but we leave this to future work.

7 An experiment

We use sections 02-21 of CCGbank for training, section 00 for development, and section 23 for testing. The input is POS-tagged using the tagger of Ratnaparkhi (1996). However, since parsing with the new model is less efficient, only sentences ≤ 40 tokens only are used to test the model. A frequency cutoff of ≤ 20 was used to determine rare words in the training data, which are replaced with their POS-tags. Unknown words in the test data are also replaced by their POS-tags. The models are evaluated according to their Parseval scores and to the recovery of dependencies in the predicate-argument structure. Like Clark et al. (2002), we do not take the lexical category of the dependent into account, and evaluate $\langle\langle c, w\rangle, i, \langle \_, w'\rangle\rangle$ for labelled, and $\langle\langle \_, w\rangle, \_, \langle \_, w'\rangle\rangle$ for unlabelled recovery. Undirectional recovery (UdirP/UdirR) evaluates only whether there is a dependency between w and w'. Unlike unlabelled recovery, this does not pe-

[2]Beam search is as in Hockenmaier and Steedman (2002b).

nalize the parser if it mistakes a complement for an adjunct or vice versa.

In order to determine the impact of capturing different kinds of long-range dependencies, four different models were investigated: The baseline model is like the **LexCat** model of (2002b), since the structural probabilities of our model are like those of that model. **Local** only takes local dependencies into account. **LeftArgs** only takes long-range dependencies that are projected through left arguments (\X) into account. This includes for instance long-range dependencies projected by subjects, subject and object control verbs, subject extraction and left-node raising. **All** takes all long-range dependencies into account, in particular it extends **LeftArgs** by capturing also the unbounded dependencies arising through right-node-raising and object extraction. **Local**, **LeftArgs** and **All** are all tested with the aggressive beam strategy described above.

In all cases, the CCG derivation includes all long-range dependencies. However, with the models that exclude certain kinds of dependencies, it is possible that a word is conditioned on no dependencies. In these cases, the word is generated with $P(w|c)$.

Table 1 gives the performance of all four models on section 23 in terms of the accuracy of lexical categories, Parseval scores, and in terms of the recovery of word-word dependencies in the predicate-argument structure. Here, results are further broken up into the recovery of local, all long-range, bounded long-range and unbounded long-range dependencies.

	LexCat	Local	LeftArgs	All
Lex. cats:	88.2	89.9	90.1	90.1
Parseval				
LP:	76.3	78.4	78.5	78.5
LR:	75.9	78.5	79.0	78.7
UP:	82.0	83.4	83.6	83.2
UR:	81.6	83.6	83.8	83.4
Predicate-argument structure (all)				
LP:	77.3	80.8	81.6	81.5
LR:	78.2	80.6	81.5	81.4
UP:	86.4	88.3	88.9	88.7
UR:	87.4	88.1	88.8	88.6
UdirP:	88.0	89.7	90.2	90.0
UdirR:	89.0	89.5	90.1	90.0
Non-long-range dependencies				
LP:	78.9	82.5	83.0	82.9
LR:	79.5	82.3	82.7	82.6
UP:	87.5	89.7	89.9	89.8
UR:	88.1	89.4	89.6	89.4
All long-range dependencies				
LP:	60.8	62.6	67.1	66.3
LR:	64.4	63.0	68.5	68.8
UP:	75.3	74.2	78.9	78.1
UR:	80.2	74.9	80.5	80.9
Bounded long-range dependencies				
LP:	63.9	64.8	69.0	69.2
LR:	65.9	64.1	70.2	70.0
UP:	79.8	77.1	81.4	81.4
UR:	82.4	76.7	82.6	82.6
Unbounded long-range dependencies				
LP:	46.0	50.4	55.6	52.4
LR:	54.7	55.8	58.7	61.2
UP:	54.1	58.2	63.8	61.1
UR:	66.5	63.7	66.8	69.9

Table 1: Evaluation (sec. 23, ≤ 40 words).

LexCat does not capture any word-word dependencies. Its performance on the recovery of predicate-argument structure can be improved by 3% by capturing only local word-word dependencies (**Local**). This excludes certain kinds of dependencies that were captured by the SD model. For instance, the dependency between the head of a noun phrase and the head of a reduced relative clause (*the shares bought by John*) is captured by the SD model, since *shares* and *bought* are both heads of the local trees that are combined to form the complex noun phrase. However, in the SD model the probability of this dependency can only be estimated from occurrences of the same construction, since dependency relations are defined in terms of local trees and not in terms of the underlying predicate-argument structure. By including long-range dependencies on left arguments (such as subjects) (**LeftArgs**), a further improvement of 0.7% on the recovery of predicate-argument structure is obtained. This model captures the dependency between *shares* and *bought*. In contrast to the SD model, it can use all instances of *shares* as the subject of a passive verb in the training data to estimate this probability. Therefore, even if *shares* and *bought* do not co-occur in this particular construction in the training data, the event that is modelled by our dependency model might not be unseen, since it could have occurred in another syntactic context.

Our results indicate that in order to perform well on long-range dependencies, they have to be included in the model, since **Local**, the model that captures only local dependencies performs worse on long-range dependencies than **LexCat**, the model that captures no word-word dependencies. However, with more than 5% difference on labelled precision and recall on long-range dependencies, the model which captures long-range dependencies on left arguments performs significantly better on recovering long-range dependencies than **Local**. The greatest difference in performance between the models which do capture long-range dependencies and the models which do not is on long-range dependencies. This indicates that, at least in the kind of model considered here, it is very important to model not just local, but also long-range dependencies. It is not clear why **All**, the model that includes all dependencies, performs slightly worse than the model which includes only long-range dependencies on subjects.

On the Wall Street Journal task, the overall performance of this model is lower than that of the SD model of Hockenmaier and Steedman (2002b). In that model, words are generated at the maximal projection of constituents; therefore, the structural probabilities can also be conditioned on words, which improves the scores by about 2%. It is also very likely that the performance of the new models is harmed by the very aggressive beam search.

8 Conclusion and future work

This paper has defined a new generative model for CCG derivations which captures the word-word dependencies in the corresponding predicate-argument structure, including bounded and unbounded long-range dependencies. In contrast to the conditional model of Clark et al. (2002), our model captures these dependencies in a sound and consistent manner. The experiments presented here demonstrate that the performance of a simple baseline model can be improved significantly if long-range dependencies are also captured. In particular, our results indicate that it is important not to restrict the model to local dependencies. Future work will address the question whether these models can be run with a less aggressive beam search strategy, or whether a different parsing algorithm is more suitable. The problems that arise due to the overly aggressive beam search strategy might be overcome if we used an n-best parser with a simpler probability model (eg. of the kind proposed by Hockenmaier and Steedman (2002b)) and used the new model as a re-ranker. The current implementation uses a very simple method of estimating the probabilities of multiple dependencies, and more sophisticated techniques should be investigated.

We have argued that a model of the kind proposed in this paper is essential for parsing languages with freer word order, such as Dutch or Czech, where the model of Hockenmaier and Steedman (2002b) (and other models of surface dependencies) would systematically capture the wrong dependencies, even if only local dependencies are taken into account. For English, our experimental results demonstrate that our model benefits greatly from modelling not only local, but also long-range dependencies, which are beyond the scope of surface dependency models.

Acknowledgements

I would like to thank Mark Steedman and Stephen Clark for many helpful discussions, and gratefully acknowledge support from an EPSRC studentship and grant GR/M96889, the School of Informatics, and NSF ITR grant 0205 456.

References

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the First Meeting of the NAACL*, Seattle.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building Deep Dependency Structures using a Wide-Coverage CCG Parser. In *Proceedings of the 40th Annual Meeting of the ACL.*

Michael Collins, Jan Hajic, Lance Ramshaw, and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the ACL.*

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

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring Compact Lexicalized Grammars from a Cleaner Treebank. In *Proceedings of the Third LREC*, pages 1974–1981, Las Palmas, May.

Julia Hockenmaier and Mark Steedman. 2002b. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the ACL.*

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with CCG.* Ph.D. thesis, School of Informatics, University of Edinburgh.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge Mass.