

Partial Training for a Lexicalized-Grammar Parser

Stephen Clark

Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
stephen.clark@comlab.ox.ac.uk

James R. Curran

School of Information Technologies
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Abstract

We propose a solution to the annotation bottleneck for statistical parsing, by exploiting the lexicalized nature of Combinatory Categorical Grammar (CCG). The parsing model uses predicate-argument dependencies for training, which are derived from sequences of CCG lexical categories rather than full derivations. A simple method is used for extracting dependencies from lexical category sequences, resulting in high precision, yet incomplete and noisy data. The dependency parsing model of Clark and Curran (2004b) is extended to exploit this partial training data. Remarkably, the accuracy of the parser trained on data derived from category sequences alone is only 1.3% worse in terms of F-score than the parser trained on complete dependency structures.

1 Introduction

State-of-the-art statistical parsers require large amounts of hand-annotated training data, and are typically based on the Penn Treebank, the largest treebank available for English. Even robust parsers using linguistically sophisticated formalisms, such as TAG (Chiang, 2000), CCG (Clark and Curran, 2004b; Hockenmaier, 2003), HPSG (Miyao et al., 2004) and LFG (Riezler et al., 2002; Cahill et al., 2004), often use training data derived from the Penn Treebank. The labour-intensive nature of the treebank development process, which can take many

years, creates a significant barrier for the development of parsers for new domains and languages.

Previous work has attempted parser adaptation without relying on treebank data from the new domain (Steedman et al., 2003; Lease and Charniak, 2005). In this paper we propose the use of annotated data in the new domain, but only *partially annotated* data, which reduces the annotation effort required (Hwa, 1999). We develop a parsing model which can be trained using partial data, by exploiting the properties of lexicalized grammar formalisms. The formalism we use is Combinatory Categorical Grammar (Steedman, 2000), together with a parsing model described in Clark and Curran (2004b) which we adapt for use with partial data.

Parsing with Combinatory Categorical Grammar (CCG) takes place in two stages: first, CCG *lexical categories* are assigned to the words in the sentence, and then the categories are combined by the parser (Clark and Curran, 2004a). The lexical categories can be thought of as detailed part of speech tags and typically express subcategorization information. We exploit the fact that CCG lexical categories contain a lot of syntactic information, and can therefore be used for training a full parser, even though attachment information is not explicitly represented in a category sequence. Our partial training regime only requires sentences to be annotated with lexical categories, rather than full parse trees; therefore the data can be produced much more quickly for a new domain or language (Clark et al., 2004).

The partial training method uses the log-linear dependency model described in Clark and Curran (2004b), which uses sets of predicate-argument de-

dependencies, rather than derivations, for training. Our novel idea is that, since there is so much information in the lexical category sequence, most of the correct dependencies can be easily inferred from the categories alone. More specifically, for a given sentence and lexical category sequence, we train on those predicate-argument dependencies *which occur in $k\%$ of the derivations licenced by the lexical categories*. By setting the k parameter high, we can produce a set of high precision dependencies for training. A similar idea is proposed by Carroll and Briscoe (2002) for producing high precision data for lexical acquisition.

Using this procedure we are able to produce dependency data with over 99% precision and, remarkably, up to 86% recall, when compared against the complete gold-standard dependency data. The high recall figure results from the significant amount of syntactic information in the lexical categories, which reduces the ambiguity in the possible dependency structures. Since the recall is not 100%, we require a log-linear training method which works with partial data. Riezler et al. (2002) describe a partial training method for a log-linear LFG parsing model in which the “correct” LFG derivations for a sentence are those consistent with the less detailed gold standard derivation from the Penn Treebank. We use a similar method here by treating a CCG derivation as correct if it is *consistent with* the high-precision partial dependency structure. Section 3 explains what we mean by consistency in this context.

Surprisingly, the accuracy of the parser trained on partial data approaches that of the parser trained on full data: our best partial-data model is only 1.3% worse in terms of dependency F-score than the full-data model, despite the fact that the partial data does not contain *any* explicit attachment information.

2 The CCG Parsing Model

Clark and Curran (2004b) describes two log-linear parsing models for CCG: a normal-form derivation model and a dependency model. In this paper we use the dependency model, which requires sets of predicate-argument dependencies for training.¹

¹Hockenmaier and Steedman (2002) describe a generative model of normal-form derivations; one possibility for training this model on partial data, which has not been explored, is to use the EM algorithm (Pereira and Schabes, 1992).

The predicate-argument dependencies are represented as 5-tuples: $\langle h_f, f, s, h_a, l \rangle$, where h_f is the lexical item of the lexical category expressing the dependency relation; f is the lexical category; s is the argument slot; h_a is the head word of the argument; and l encodes whether the dependency is non-local. For example, the dependency encoding *company* as the object of *bought* (as in *IBM bought the company*) is represented as follows:

$$\langle \text{bought}_2, (S \setminus NP_1) / NP_2, 2, \text{company}_4, - \rangle \quad (1)$$

CCG dependency structures are sets of predicate-argument dependencies. We define the probability of a dependency structure as the sum of the probabilities of all those derivations leading to that structure (Clark and Curran, 2004b). “Spurious ambiguity” in CCG means that there can be more than one derivation leading to any one dependency structure. Thus, the probability of a dependency structure, π , given a sentence, S , is defined as follows:

$$P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S) \quad (2)$$

where $\Delta(\pi)$ is the set of derivations which lead to π .

The probability of a $\langle d, \pi \rangle$ pair, ω , conditional on a sentence S , is defined using a log-linear form:

$$P(\omega|S) = \frac{1}{Z_S} e^{\lambda \cdot \mathbf{f}(\omega)} \quad (3)$$

where $\lambda \cdot \mathbf{f}(\omega) = \sum_i \lambda_i f_i(\omega)$. The function f_i is the integer-valued frequency function of the i th feature; λ_i is the weight of the i th feature; and Z_S is a normalising constant.

Clark and Curran (2004b) describes the training procedure for the dependency model, which uses a discriminative estimation method by maximising the conditional likelihood of the model given the data (Riezler et al., 2002). The optimisation of the objective function is performed using the limited-memory BFGS numerical optimisation algorithm (Nocedal and Wright, 1999; Malouf, 2002), which requires calculation of the objective function and the gradient of the objective function at each iteration.

The objective function is defined below, where $L(\Lambda)$ is the likelihood and $G(\Lambda)$ is a Gaussian prior term for smoothing.

He anticipates growth for the auto maker
 \overline{NP} $\overline{(S[dcI]\backslash NP)/NP}$ \overline{NP} $\overline{(NP\backslash NP)/NP}$ $\overline{NP[nb]/N}$ $\overline{N/N}$ \overline{N}

Figure 1: Example sentence with CCG lexical categories

$$\begin{aligned}
L'(\Lambda) &= L(\Lambda) - G(\Lambda) \\
&= \sum_{j=1}^m \log \sum_{d \in \Delta(\pi_j)} e^{\lambda \cdot \mathbf{f}(d, \pi_j)} \\
&\quad - \sum_{j=1}^m \log \sum_{\omega \in \rho(S_j)} e^{\lambda \cdot \mathbf{f}(\omega)} - \sum_{i=1}^n \frac{\lambda_i^2}{2\sigma^2}
\end{aligned} \tag{4}$$

S_1, \dots, S_m are the sentences in the training data; π_1, \dots, π_m are the corresponding gold-standard dependency structures; $\rho(S)$ is the set of possible \langle derivation, dependency-structure \rangle pairs for S ; σ is a smoothing parameter; and n is the number of features. The components of the gradient vector are:

$$\begin{aligned}
\frac{\partial L'(\Lambda)}{\partial \lambda_i} &= \sum_{j=1}^m \sum_{d \in \Delta(\pi_j)} \frac{e^{\lambda \cdot \mathbf{f}(d, \pi_j)} f_i(d, \pi_j)}{\sum_{d \in \Delta(\pi_j)} e^{\lambda \cdot \mathbf{f}(d, \pi_j)}} \\
&\quad - \sum_{j=1}^m \sum_{\omega \in \rho(S_j)} \frac{e^{\lambda \cdot \mathbf{f}(\omega)} f_i(\omega)}{\sum_{\omega \in \rho(S_j)} e^{\lambda \cdot \mathbf{f}(\omega)}} - \frac{\lambda_i}{\sigma^2}
\end{aligned} \tag{5}$$

The first two terms of the gradient are expectations of feature f_i : the first expectation is over all derivations leading to each gold-standard dependency structure, and the second is over all derivations for each sentence in the training data. The estimation process attempts to make the expectations in (5) equal (ignoring the Gaussian prior term). Another way to think of the estimation process is that it attempts to put as much mass as possible on the derivations leading to the gold-standard structures (Riezler et al., 2002).

Calculation of the feature expectations requires summing over all derivations for a sentence, and summing over all derivations leading to a gold-standard dependency structure. Clark and Curran (2003) shows how the sum over the complete derivation space can be performed efficiently using a packed chart and the inside-outside algorithm, and Clark and Curran (2004b) extends this method to sum over all derivations leading to a gold-standard dependency structure.

3 Partial Training

The partial data we use for training the dependency model is derived from CCG lexical category sequences only. Figure 1 gives an example sentence adapted from CCGbank (Hockenmaier, 2003) together with its lexical category sequence. Note that, although the attachment of the prepositional phrase to the noun phrase is not explicitly represented, it can be inferred in this example because the lexical category assigned to the preposition has to combine with a noun phrase to the left, and in this example there is only one possibility. One of the key insights in this paper is that the significant amount of syntactic information in CCG lexical categories allows us to infer attachment information in many cases.

The procedure we use for extracting dependencies from a sequence of lexical categories is to return all those dependencies which occur in $k\%$ of the derivations licenced by the categories. By giving the k parameter a high value, we can extract sets of dependencies with very high precision; in fact, assuming that the correct lexical category sequence licences the correct derivation, setting k to 100 must result in 100% precision, since any dependency which occurs in every derivation must occur in the correct derivation. Of course the recall is not guaranteed to be high; decreasing k has the effect of increasing recall, but at the cost of decreasing precision.

The training method described in Section 2 can be adapted to use the (potentially incomplete) sets of dependencies returned by our extraction procedure. In Section 2 a derivation was considered correct if it produced the complete set of gold-standard dependencies. In our partial-data version a derivation is considered correct if it produces dependencies which are *consistent with* the dependencies returned by our extraction procedure. We define consistency as follows: a set of dependencies D is consistent with a set G if G is a subset of D . We also say that a derivation d is consistent with dependency set G if G is a subset of the dependencies produced by d .

This definition of “correct derivation” will introduce some noise into the training data. Noise arises from sentences where the recall of the extracted dependencies is less than 100%, since some of the derivations which are consistent with the extracted dependencies for such sentences will be incorrect. Noise also arises from sentences where the precision of the extracted dependencies is less than 100%, since for these sentences every derivation which is consistent with the extracted dependencies will be incorrect. The hope is that, if an incorrect derivation produces mostly correct dependencies, then it can still be useful for training. Section 4 shows how the precision and recall of the extracted dependencies varies with k and how this affects parsing accuracy.

The definitions of the objective function (4) and the gradient (5) for training remain the same in the partial-data case; the only differences are that $\Delta(\pi)$ is now defined to be those derivations which are consistent with the partial dependency structure π , and the gold-standard dependency structures π_j are the partial structures extracted from the gold-standard lexical category sequences.²

Clark and Curran (2004b) gives an algorithm for finding all derivations in a packed chart which produce a particular set of dependencies. This algorithm is required for calculating the value of the objective function (4) and the first feature expectation in (5). We adapt this algorithm for finding all derivations which are consistent with a partial dependency structure. The new algorithm is shown in Figure 2.

The algorithm relies on the definition of a packed chart, which is an instance of a *feature forest* (Miyao and Tsujii, 2002). The idea behind a packed chart is that equivalent chart entries of the same type and in the same cell are grouped together, and back pointers to the daughters indicate how an individual entry was created. Equivalent entries form the same structures in any subsequent parsing.

A feature forest is defined in terms of *disjunctive* and *conjunctive* nodes. For a packed chart, the individual entries in a cell are conjunctive nodes, and the equivalence classes of entries are disjunctive nodes. The definition of a feature forest is as follows:

A *feature forest* Φ is a tuple $\langle C, D, R, \gamma, \delta \rangle$ where:

$\langle C, D, R, \gamma, \delta \rangle$ is a packed chart / feature forest
 G is a set of dependencies returned by the extraction procedure
 Let c be a conjunctive node
 Let d be a disjunctive node
 $deps(c)$ is the set of dependencies on node c
 $cdeps(c) = |deps(c) \cap G|$
 $dmax(c) = \sum_{d \in \delta(c)} dmax(d) + cdeps(c)$
 $dmax(d) = \max\{dmax(c) \mid c \in \gamma(d)\}$
mark(d):
 mark d as a correct node
foreach $c \in \gamma(d)$
 if $dmax(c) == dmax(d)$
 mark c as a correct node
foreach $d' \in \delta(c)$
mark(d')
foreach $d_r \in R$ such that $dmax(d_r) = |G|$
mark(d_r)

Figure 2: Finding nodes in derivations consistent with a partial dependency structure

- C is a set of conjunctive nodes;
- D is a set of disjunctive nodes;
- $R \subseteq D$ is a set of root disjunctive nodes;
- $\gamma : D \rightarrow 2^C$ is a conjunctive daughter function;
- $\delta : C \rightarrow 2^D$ is a disjunctive daughter function.

Dependencies are associated with conjunctive nodes in the feature forest. For example, if the disjunctive nodes (equivalence classes of individual entries) representing the categories NP and $S \setminus NP$ combine to produce a conjunctive node S , the resulting S node will have a verb-subject dependency associated with it.

In Figure 2, $cdeps(c)$ is the number of dependencies on conjunctive node c which appear in partial structure G ; $dmax(c)$ is the maximum number of dependencies in G produced by any sub-derivation headed by c ; $dmax(d)$ is the same value for disjunctive node d . Recursive definitions for calculating these values are given; the base case occurs when conjunctive nodes have no disjunctive daughters.

The algorithm identifies all those root nodes heading derivations which are consistent with the partial dependency structure G , and traverses the chart top-down marking the nodes in those derivations. The insight behind the algorithm is that, for two conjunctive nodes in the same equivalence class, if one node heads a sub-derivation producing more dependencies in G than the other node, then the node with

²Note that the procedure does return *all* the gold-standard dependencies for some sentences.

less dependencies in G cannot be part of a derivation consistent with G .

The conjunctive and disjunctive nodes appearing in derivations consistent with G form a new “gold-standard” feature forest. The gold-standard forest, and the complete forest containing all derivations spanning the sentence, can be used to estimate the likelihood value and feature expectations required by the estimation algorithm. Let $E_{\Lambda}^{\Phi} f_i$ be the expected value of f_i over the forest Φ for model Λ ; then the values in (5) can be obtained by calculating $E_{\Lambda}^{\Phi_j} f_i$ for the complete forest Φ_j for each sentence S_j in the training data (the second sum in (5)), and also $E_{\Lambda}^{\Psi_j} f_i$ for each forest Ψ_j of derivations consistent with the partial gold-standard dependency structure for sentence S_j (the first sum in (5)):

$$\frac{\partial L(\Lambda)}{\partial \lambda_i} = \sum_{j=1}^m (E_{\Lambda}^{\Psi_j} f_i - E_{\Lambda}^{\Phi_j} f_i) \quad (6)$$

The likelihood in (4) can be calculated as follows:

$$L(\Lambda) = \sum_{j=1}^m (\log Z_{\Psi_j} - \log Z_{\Phi_j}) \quad (7)$$

where $\log Z_{\Phi}$ is the normalisation constant for Φ .

4 Experiments

The resource used for the experiments is CCGbank (Hockenmaier, 2003), which consists of normal-form CCG derivations derived from the phrase-structure trees in the Penn Treebank. It also contains predicate-argument dependencies which we use for development and final evaluation.

4.1 Accuracy of Dependency Extraction

Sections 2-21 of CCGbank were used to investigate the accuracy of the partial dependency structures returned by the extraction procedure. Full, correct dependency structures for the sentences in 2-21 were created by running our CCG parser (Clark and Curran, 2004b) over the gold-standard derivation for each sentence, outputting the dependencies. This resulted in full dependency structures for 37,283 of the sentences in sections 2-21.

Table 1 gives precision and recall values for the dependencies obtained from the extraction procedure, for the 37,283 sentences for which we have

k	Precision	Recall	SentAcc
0.99999	99.76	74.96	13.84
0.9	99.69	79.37	16.52
0.85	99.65	81.30	18.40
0.8	99.57	82.96	19.51
0.7	99.09	85.87	22.46
0.6	98.00	88.67	26.28

Table 1: Accuracy of the Partial Dependency Data

complete dependency structures. The SentAcc column gives the percentage of training sentences for which the partial dependency structures are completely correct. For a given sentence, the extraction procedure returns all dependencies occurring in at least $k\%$ of the derivations licenced by the gold-standard lexical category sequence. The lexical category sequences for the sentences in 2-21 can easily be read off the CCGbank derivations.

The derivations licenced by a lexical category sequence were created using the CCG parser described in Clark and Curran (2004b). The parser uses a small number of combinatory rules to combine the categories, along with the CKY chart-parsing algorithm described in Steedman (2000). It also uses some unary type-changing rules and punctuation rules obtained from the derivations in CCGbank.³ The parser builds a packed representation, and counting the number of derivations in which a dependency occurs can be performed using a dynamic programming algorithm similar to the inside-outside algorithm.

Table 1 shows that, by varying the value of k , it is possible to get the recall of the extracted dependencies as high as 85.9%, while still maintaining a precision value of over 99%.

4.2 Accuracy of the Parser

The training data for the dependency model was created by first supertagging the sentences in sections 2-21, using the supertagger described in Clark and Curran (2004b).⁴ The average number of categories

³Since our training method is intended to be applicable in the absence of derivation data, the use of such rules may appear suspect. However, we argue that the type-changing and punctuation rules could be manually created for a new domain by examining the lexical category data.

⁴An improved version of the supertagger was used for this paper in which the forward-backward algorithm is used to calculate the lexical category probability distributions.

assigned to each word is determined by a parameter, β , in the supertagger. A category is assigned to a word if the category’s probability is within β of the highest probability category for that word.

For these experiments, we used a β value of 0.01, which assigns roughly 1.6 categories to each word, on average; we also ensured that the correct lexical category was in the set assigned to each word. (We did not do this when parsing the test data.) For some sentences, the packed charts can become very large. The supertagging approach we adopt for training differs to that used for testing: if the size of the chart exceeds some threshold, the value of β is increased, reducing ambiguity, and the sentence is supertagged and parsed again. The threshold which limits the size of the charts was set at 300 000 individual entries. Two further values of β were used: 0.05 and 0.1.

Packed charts were created for each sentence and stored in memory. It is essential that the packed charts for each sentence contain at least one derivation leading to the gold-standard dependency structure. Not all rule instantiations in CCGbank can be produced by our parser; hence it is not possible to produce the gold standard for every sentence in Sections 2-21. For the full-data model we used 34 336 sentences (86.7% of the total). For the partial-data models we were able to use slightly more, since the partial structures are easier to produce. Here we used 35,709 sentences ($k = 0.85$).

Since some of the packed charts are very large, we used an 18-node Beowulf cluster, together with a parallel version of the BFGS training algorithm. The training time and number of iterations to convergence were 172 minutes and 997 iterations for the full-data model, and 151 minutes and 861 iterations for the partial-data model ($k = 0.85$). Approximate memory usage in each case was 17.6 GB of RAM.

The dependency model uses the same set of features described in Clark and Curran (2004b): dependency features representing predicate-argument dependencies (with and without distance measures); rule instantiation features encoding the combining categories together with the result category (with and without a lexical head); lexical category features, consisting of word–category pairs at the leaf nodes; and root category features, consisting of headword–category pairs at the root nodes. Further

k	LP	LR	F	CatAcc
0.99999	85.80	84.51	85.15	93.77
0.9	85.86	84.51	85.18	93.78
0.85	85.89	84.50	85.19	93.71
0.8	85.89	84.45	85.17	93.70
0.7	85.52	84.07	84.79	93.72
0.6	84.99	83.70	84.34	93.65
FullData	87.16	85.84	86.50	93.79
Random	74.63	72.53	73.57	89.31

Table 2: Accuracy of the Parser on Section 00

generalised features for each feature type are formed by replacing words with their POS tags.

Only features which occur more than once in the training data are included, except that the cutoff for the rule features is 10 or more and the counting is performed across all derivations licenced by the gold-standard lexical category sequences. The larger cutoff was used since the productivity of the grammar can lead to large numbers of these features. The dependency model has 548 590 features. In order to provide a fair comparison, the same feature set was used for the partial-data and full-data models.

The CCG parsing consists of two phases: first the supertagger assigns the most probable categories to each word, and then the small number of combinatory rules, plus the type-changing and punctuation rules, are used with the CKY algorithm to build a packed chart.⁵ We use the method described in Clark and Curran (2004b) for integrating the supertagger with the parser: initially a small number of categories is assigned to each word, and more categories are requested if the parser cannot find a spanning analysis. The “maximum-recall” algorithm described in Clark and Curran (2004b) is used to find the highest scoring dependency structure.

Table 2 gives the accuracy of the parser on Section 00 of CCGbank, evaluated against the predicate-argument dependencies in CCGbank.⁶ The table gives labelled precision, labelled recall and F-score, and lexical category accuracy. Numbers are given for the partial-data model with various values of k , and for the full-data model, which provides an up-

⁵Gold-standard POS tags from CCGbank were used for all the experiments in this paper.

⁶There are some dependency types produced by our parser which are not in CCGbank; these were ignored for evaluation.

	LP	LR	F	CatAcc
$k = 0.85$	86.21	85.01	85.60	93.90
FullData	87.50	86.37	86.93	94.01

Table 3: Accuracy of the Parser on Section 23

k	Precision	Recall	SentAcc
0.99999	99.71	80.16	17.48
0.9999	99.68	82.09	19.13
0.999	99.49	85.18	22.18
0.99	99.00	88.95	27.69
0.95	98.34	91.69	34.95
0.9	97.82	92.84	39.18

Table 4: Accuracy of the Partial Dependency Data using Inside-Outside Scores

per bound for the partial-data model. We also give a lower bound which we obtain by randomly traversing a packed chart top-down, giving equal probability to each conjunctive node in an equivalence class. The precision and recall figures are over those sentences for which the parser returned an analysis (99.27% of Section 00).

The best result is obtained for a k value of 0.85, which produces partial dependency data with a precision of 99.7 and a recall of 81.3. Interestingly, the results show that decreasing k further, which results in partial data with a higher recall and only a slight loss in precision, harms the accuracy of the parser. The Random result also dispels any suspicion that the partial-model is performing well simply because of the supertagger; clearly there is still much work to be done after the supertagging phase.

Table 3 gives the accuracy of the parser on Section 23, using the best performing partial-data model on Section 00. The precision and recall figures are over those sentences for which the parser returned an analysis (99.63% of Section 23). The results show that the partial-data model is only 1.3% F-score short of the upper bound.

4.3 Further Experiments with Inside-Outside

In a final experiment, we attempted to exploit the high accuracy of the partial-data model by using it to provide new training data. For each sentence in Section 2-21, we parsed the gold-standard lexical category sequences and used the best performing

partial-data model to assign scores to each dependency in the packed chart. The score for a dependency was the sum of the probabilities of all derivations producing that dependency, which can be calculated using the inside-outside algorithm. (This is the score used by the maximum-recall parsing algorithm.) Partial dependency structures were then created by returning all dependencies whose score was above some threshold k , as before. Table 4 gives the accuracy of the data created by this procedure. Note how these values differ to those reported in Table 1.

We then trained the dependency model on this partial data using the same method as before. However, the performance of the parser on Section 00 using these new models was below that of the previous best performing partial-data model for all values of k . We report this negative result because we had hypothesised that using a probability model to score the dependencies, rather than simply the number of derivations in which they occur, would lead to improved performance.

5 Conclusions

Our main result is that it is possible to train a CCG dependency model from lexical category sequences alone and still obtain parsing results which are only 1.3% worse in terms of labelled F-score than a model trained on complete data. This is a noteworthy result and demonstrates the significant amount of information encoded in CCG lexical categories.

The engineering implication is that, since the dependency model can be trained without annotating recursive structures, and only needs sequence information at the word level, then it can be ported rapidly to a new domain (or language) by annotating new sequence data in that domain.

One possible response to this argument is that, since the lexical category sequence contains so much syntactic information, then the task of annotating category sequences must be almost as labour intensive as annotating full derivations. To test this hypothesis fully would require suitable annotation tools and subjects skilled in CCG annotation, which we do not currently have access to.

However, there is some evidence that annotating category sequences can be done very efficiently. Clark et al. (2004) describes a porting experiment

in which a CCG parser is adapted for the question domain. The supertagger component of the parser is trained on questions annotated at the lexical category level only. The training data consists of over 1,000 annotated questions which took less than a week to create. This suggests, as a very rough approximation, that 4 annotators could annotate 40,000 sentences with lexical categories (the size of the Penn Treebank) in a few months.

Another advantage of annotating with lexical categories is that a CCG supertagger can be used to perform most of the annotation, with the human annotator only required to correct the mistakes made by the supertagger. An accurate supertagger can be bootstrapped quickly, leaving only a small number of corrections for the annotator. A similar procedure is suggested by Doran et al. (1997) for porting an LTAG grammar to a new domain.

We have a proposed a novel solution to the annotation bottleneck for statistical parsing which exploits the lexicalized nature of CCG, and may therefore be applicable to other lexicalized grammar formalisms such as LTAG.

References

- A. Cahill, M. Burke, R. O'Donovan, J. van Genabith, and A. Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the ACL*, pages 320–327, Barcelona, Spain.
- John Carroll and Ted Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 134–140, Taipei, Taiwan.
- David Chiang. 2000. Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th Meeting of the ACL*, pages 456–463, Hong Kong.
- Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the EMNLP Conference*, pages 97–104, Sapporo, Japan.
- Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, pages 282–288, Geneva, Switzerland.
- Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, pages 104–111, Barcelona, Spain.
- Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the EMNLP Conference*, pages 111–118, Barcelona, Spain.
- C. Doran, B. Hockey, P. Hopely, J. Rosenzweig, A. Sarkar, B. Srinivas, F. Xia, A. Nasr, and O. Rambow. 1997. Maintaining the forest and burning out the underbrush in XTAG. In *Proceedings of the ENVGRAM Workshop*, Madrid, Spain.
- Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorical Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.
- Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Meeting of the ACL*, pages 73–79, University of Maryland, MD.
- Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea.
- Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.
- Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.
- Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684–693, Hainan Island, China.
- Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York, USA.
- Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Meeting of the ACL*, pages 128–135, Newark, DE.
- Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the ACL*, pages 271–278, Philadelphia, PA.
- Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steve Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 11th Conference of the European Association for Computational Linguistics*, Budapest, Hungary.
- Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.