Parsing the WSJ using CCG and Log-Linear Models

Stephen Clark
School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh, UK
stephen.clark@ed.ac.uk

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

Abstract

This paper describes and evaluates log-linear parsing models for Combinatory Categorial Grammar (CCG). A parallel implementation of the L-BFGS optimisation algorithm is described, which runs on a Beowulf cluster allowing the complete Penn Treebank to be used for estimation. We also develop a new efficient parsing algorithm for CCG which maximises expected recall of dependencies. We compare models which use all CCG derivations, including non-standard derivations, with normal-form models. The performances of the two models are comparable and the results are competitive with existing wide-coverage CCG parsers.

1 Introduction

A number of statistical parsing models have recently been developed for Combinatory Categorial Grammar (CCG; Steedman, 2000) and used in parsers applied to the WSJ Penn Treebank (Clark et al., 2002; Hockenmaier and Steedman, 2002; Hockenmaier, 2003b). In Clark and Curran (2003) we argued for the use of log-linear parsing models for CCG. However, estimating a log-linear model for a wide-coverage CCG grammar is very computationally expensive. Following Miyao and Tsujii (2002), we showed how the estimation can be performed efficiently by applying the inside-outside algorithm to a packed chart. We also showed how the complete WSJ Penn Treebank can be used for training by developing a parallel version of Generalised Iterative Scaling (GIS) to perform the estimation.

This paper significantly extends our earlier work in a number of ways. First, we evaluate a number of log-linear models, obtaining results which are competitive with the state-of-the-art for CCG parsing. We also compare log-linear models which use all CCG derivations, including non-standard derivations, with normal-form models. Second, we find that GIS is unsuitable for estimating a model of the size being considered, and develop a parallel version of the L-BFGS algorithm (Nocedal and Wright, 1999). And finally, we show that the parsing algorithm described in Clark and Curran (2003) is extremely slow in some cases, and suggest an efficient alternative based on Goodman (1996).

The development of parsing and estimation algorithms for models which use all derivations extends existing CCG parsing techniques, and allows us to test whether there is useful information in the additional derivations. However, we find that the performance of the normal-form model is at least as good as the all-derivations model, in our experiments to-date. The normal-form approach allows the use of additional constraints on rule applications, leading to a smaller model, reducing the computational resources required for estimation, and resulting in an extremely efficient parser.

This paper assumes a basic understanding of CCG; see Steedman (2000) for an introduction, and Clark et al. (2002) and Hockenmaier (2003a) for an introduction to statistical parsing with CCG.

2 Parsing Models for CCG

CCG is unusual among grammar formalisms in that, for each *derived structure* for a sentence, there can be many derivations leading to that structure. The presence of such ambiguity, sometimes referred to as *spurious ambiguity*, enables CCG to produce elegant analyses of coordination and extraction phenomena (Steedman, 2000). However, the introduction of extra derivations increases the complexity of the modelling and parsing problem.

Clark et al. (2002) handle the additional derivations by modelling the derived structure, in their case dependency structures. They use a conditional model, based on Collins (1996), which, as the authors acknowledge, has a number of theoretical deficiencies; thus the results of Clark et al. provide a useful baseline for the new models presented here.

Hockenmaier (2003a) uses a model which favours only one of the derivations leading to a derived structure, namely the *normal-form* derivation (Eisner, 1996). In this paper we compare the normal-form approach with a dependency model. For the dependency model, we define the probability of a dependency structure as follows:

$$P(\pi|S) = \sum_{d \in \Delta(\pi)} P(d, \pi|S) \tag{1}$$

where π is a dependency structure, S is a sentence and $\Delta(\pi)$ is the set of derivations which lead to π. This extends the approach of Clark et al. (2002) who modelled the dependency structures directly, not using any information from the derivations. In contrast to the dependency model, the normal-form model simply defines a distribution over normal-form derivations.

The dependency structures considered in this paper are described in detail in Clark et al. (2002) and Clark and Curran (2003). Each argument slot in a CCG lexical category represents a dependency relation, and a dependency is defined as a 5-tuple $\langle h_f, f, s, h_a, l\rangle$, where h_f is the head word of the lexical category, f is the lexical category, s is the argument slot, h_a is the head word of the argument, and l indicates whether the dependency is long-range. For example, the long-range dependency encoding *company* as the extracted object of *bought* (as in *the company that IBM bought*) is represented as the following 5-tuple:

$$\langle \textit{bought}, (S[\textit{dcl}]\backslash NP_1)/NP_2, 2, \textit{company}, *\rangle$$

where $*$ is the category $(NP\backslash NP)/(S[\textit{dcl}]/NP)$ assigned to the relative pronoun. For local dependencies l is assigned a null value. A dependency structure is a multiset of these dependencies.

3 Log-Linear Parsing Models

Log-linear models (also known as Maximum Entropy models) are popular in NLP because of the ease with which discriminating features can be included in the model. Log-linear models have been applied to the parsing problem across a range of grammar formalisms, e.g. Riezler et al. (2002) and Toutanova et al. (2002). One motivation for using a log-linear model is that long-range dependencies which CCG was designed to handle can easily be encoded as features.

A conditional log-linear model of a parse $\omega \in \Omega$, given a sentence S, is defined as follows:

$$P(\omega|S) = \frac{1}{Z_S} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)} \tag{2}$$

where $\boldsymbol{\lambda}.\boldsymbol{f}(\omega) = \sum_i \lambda_i f_i(\omega)$. The function f_i is a *feature* of the parse which can be any real-valued function over the space of parses Ω. Each feature f_i has an associated *weight* λ_i which is a parameter of the model to be estimated. Z_S is a normalising constant which ensures that $P(\omega|S)$ is a probability distribution:

$$Z_S = \sum_{\omega' \in \rho(S)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega')} \tag{3}$$

where $\rho(S)$ is the set of possible parses for S.

For the dependency model a parse, ω, is a $\langle d, \pi\rangle$ pair (as given in (1)). A feature is a count of the number of times some configuration occurs in d or the number of times some dependency occurs in π. Section 6 gives examples of features.

3.1 The Dependency Model

We follow Riezler et al. (2002) in using a discriminative estimation method by maximising the conditional likelihood of the model given the data. For the dependency model, the data consists of sentences $S_1, \ldots, S_m$, together with gold standard dependency structures, $\pi_1, \ldots, \pi_m$. The gold standard structures are multisets of dependencies, as described earlier. Section 6 explains how the gold standard structures are obtained.

The objective function of a model Λ is the conditional log-likelihood, $L(\Lambda)$, minus a Gaussian prior term, $G(\Lambda)$, used to reduce overfitting (Chen and Rosenfeld, 1999). Hence, given the definition of the probability of a dependency structure (1), the objective function is as follows:

$$\begin{aligned} L'(\Lambda) &= L(\Lambda) - G(\Lambda) \qquad (4)\\ &= \log \prod_{j=1}^{m} P_\Lambda(\pi_j|S_j) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\ &= \sum_{j=1}^{m} \log \frac{\sum_{d\in\Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)}}{\sum_{\omega\in\rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)}} - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\ &= \sum_{j=1}^{m} \log \sum_{d\in\Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)} \\ &\quad - \sum_{j=1}^{m} \log \sum_{\omega\in\rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)} - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \end{aligned}$$

where n is the number of features. Rather than have a different smoothing parameter σ_i for each feature, we use a single parameter σ.

We use a technique from the numerical optimisation literature, the L-BFGS algorithm (Nocedal and Wright, 1999), to optimise the objective function. L-BFGS is an iterative algorithm which requires the gradient of the objective function to be computed at each iteration. The components of the gradient vector are as follows:

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

The first two terms in (5) are expectations of feature f_i: the first expectation is over all derivations leading to each gold standard dependency structure; the second is over all derivations for each sentence in the training data. Setting the gradient to zero yields the usual maximum entropy constraints (Berger et al., 1996), except that in this case the empirical values are themselves expectations (over all derivations leading to each gold standard dependency structure). The estimation process attempts to make the expectations equal, by putting as much mass as possible on the derivations leading to the gold standard structures.[1] The Gaussian prior term penalises any model whose weights get too large in absolute value.

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. In both cases there can be exponentially many derivations, and so enumerating all derivations is not possible (at least for wide-coverage automatically extracted grammars). Clark and Curran (2003) show how the sum over the complete derivation space can be performed efficiently using a packed chart and a variant of the inside-outside algorithm. Section 5 shows how the same technique can also be applied to all derivations leading to a gold standard dependency structure.

3.2 The Normal-Form Model

The objective function and gradient vector for the normal-form model are as follows:

$$\begin{aligned}L'(\Lambda) &= L(\Lambda) - G(\Lambda) \\ &= \log \prod_{j=1}^{m} P_\Lambda(d_j|S_j) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2}\end{aligned} \tag{6}$$

$$\begin{aligned}\frac{\partial L'(\Lambda)}{\partial \lambda_i} &= \sum_{j=1}^{m} f_i(d_j) \\ &\quad - \sum_{j=1}^{m} \sum_{d \in \theta(S_j)} \frac{e^{\boldsymbol{\lambda} . \boldsymbol{f}(d)} f_i(d)}{\sum_{d \in \theta(S_j)} e^{\boldsymbol{\lambda} . \boldsymbol{f}(d)}} - \frac{\lambda_i}{\sigma_i^2}\end{aligned} \tag{7}$$

where d_j is the the gold standard derivation for sentence S_j and $\theta(S_j)$ is the set of possible derivations for S_j. Note that the empirical expectation in (7) is simply a count of the number of times the feature appears in the gold-standard derivations.

4 Packed Charts

The packed charts perform a number of roles: they are a compact representation of a very large number of CCG derivations; they allow recovery of the highest scoring parse or dependency structure without enumerating all derivations; and they represent an instance of what Miyao and Tsujii (2002) call a *feature forest*, which is used to efficiently estimate a log-linear model. The idea behind a packed chart is simple: equivalent chart entries of the same type, 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.

Since the packed charts are used for model estimation and recovery of the highest scoring parse or dependency structure, the features in the model partly determine which entries can be grouped together. In this paper we use features from the dependency structure, and features defined on the local rule instantiations.[2] Hence, any two entries with identical category type, identical head, and identical *unfilled* dependencies are equivalent. Note that not all features are local to a rule instantiation; for example, features encoding long-range dependencies may involve words which are a long way apart in the sentence.

For the purposes of estimation and finding the highest scoring parse or dependency structure, only entries which are part of a derivation spanning the whole sentence are relevant. These entries can be easily found by traversing the chart top-down, starting with the entries which span the sentence. The entries within spanning derivations form a feature forest (Miyao and Tsujii, 2002). A *feature forest* Φ is a tuple $\langle C, D, R, \gamma, \delta \rangle$ where:

- 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.

The individual entries in a cell are conjunctive nodes, and the equivalence classes of entries are dis-

[1] See Riezler et al. (2002) for a similar description in the context of LFG parsing.

[2] By *rule instantiation* we mean the local tree arising from the application of a CCG combinatory rule.

$\langle C, D, R, \gamma, \delta \rangle$ is a packed chart / feature forest
G is a set of gold standard dependencies
Let c be a conjunctive node
Let d be a disjunctive node
$deps(c)$ is the set of dependencies on node c

$$cdeps(c) = \begin{cases} -1 & \text{if, for some } \tau \in deps(c), \tau \notin G \\ |deps(c)| & \text{otherwise} \end{cases}$$

$$dmax(c) = \begin{cases} -1 & \text{if } \; cdeps(c) = -1 \\ -1 & \text{if } \; dmax(d) = -1 \text{ for some } d \in \delta(c) \\ \sum_{d \in \delta(c)} dmax(d) + cdeps(c) & \text{otherwise} \end{cases}$$

$$dmax(d) = \max\{dmax(c) \mid c \in \gamma(d)\}$$

```
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 1: Finding nodes in correct derivations

junctive nodes. The roots of the CCG derivations represent the root disjunctive nodes.[3]

5 Efficient Estimation

The L-BFGS algorithm requires the following values at each iteration: the expected value, and the empirical expected value, of each feature (to calculate the gradient); and the value of the likelihood function. For the normal-form model, the empirical expected values and the likelihood can easily be obtained, since these only involve the single gold-standard derivation for each sentence. The expected values can be calculated using the method in Clark and Curran (2003).

For the dependency model, the computations of the empirical expected values (5) and the likelihood function (4) are more complex, since these require sums over just those derivations leading to the gold standard dependency structure. We will refer to such derivations as *correct* derivations.

Figure 1 gives an algorithm for finding nodes in a packed chart which appear in correct derivations. $cdeps(c)$ is the number of correct dependencies on conjunctive node c, and takes the value -1 if there are any incorrect dependencies on c. $dmax(c)$ is the maximum number of correct dependencies produced by any sub-derivation headed by c, and takes the value -1 if there are no sub-derivations producing only correct dependencies. $dmax(d)$ is the same value but for disjunctive node d. Recursive definitions for calculating these values are given in Figure 1; the base case occurs when conjunctive nodes have no disjunctive daughters.

The algorithm identifies all those root nodes heading derivations which produce just the correct dependencies, 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 correct dependencies than the other node (and each sub-derivation only produces correct dependencies), then the node with less correct dependencies cannot be part of a correct derivation.

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

$$\frac{\partial L(\Lambda)}{\partial \lambda_i} = \sum_{j=1}^{m} (E^{\Psi_j}_{\Lambda} f_i - E^{\Phi_j}_{\Lambda} f_i) \qquad (8)$$

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

$$L(\Lambda) = \sum_{j=1}^{m} (\log Z_{\Psi_j} - \log Z_{\Phi_j}) \qquad (9)$$

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

6 Estimation in Practice

The gold standard dependency structures are produced by running our CCG parser over the normal-form derivations in CCGbank (Hockenmaier, 2003a). Not all rule instantiations in CCGbank are instances of combinatory rules, and not all can be produced by the parser, and so gold standard structures were created for 85.5% of the sentences in sections 2-21 (33,777 sentences).

The same parser is used to produce the packed charts. The parser uses a maximum entropy supertagger (Clark and Curran, 2004) to assign lexical

[3] A more complete description of CCG feature forests is given in Clark and Curran (2003).

categories to the words in a sentence, and applies the CKY chart parsing algorithm described in Steedman (2000). For parsing the training data, we ensure that the correct category is a member of the set assigned to each word. The average number of categories assigned to each word is determined by a parameter in the supertagger. For the first set of experiments, we used a setting which assigns 1.7 categories on average per word.

The feature set for the dependency model consists of the following types of features: dependency features (with and without distance measures), rule instantiation features (with and without a lexical head), lexical category features, and root category features. Dependency features are the 5-tuples defined in Section 1. There are also three additional dependency feature types which have an extra distance field (and only include the head of the lexical category, and not the head of the argument); these count the number of words (0, 1, 2 or more), punctuation marks (0, 1, 2 or more), and verbs (0, 1 or more) between head and dependent. Lexical category features are word–category pairs at the leaf nodes, and root features are headword–category pairs at the root nodes. Rule instantiation features simply encode the combining categories together with the result category. There is an additional rule feature type which also encodes the lexical head of the resulting category. Additional generalised features for each feature type are formed by replacing words with their POS tags.

The feature set for the normal-form model is the same except that, following Hockenmaier and Steedman (2002), the dependency features are defined in terms of the local rule instantiations, by adding the heads of the combining categories to the rule instantiation features. Again there are 3 additional distance feature types, as above, which only include the head of the resulting category. We had hoped that by modelling the predicate-argument dependencies produced by the parser, rather than local rule dependencies, we would improve performance. However, using the predicate-argument dependencies in the normal-form model instead of, or in addition to, the local rule dependencies, has not led to an improvement in parsing accuracy.

Only features which occurred more than once in the training data were included, except that, for the dependency model, the cutoff for the rule features was 9 and the counting was performed across all derivations, not just the gold-standard derivation. The normal-form model has 482,007 features and the dependency model has 984,522 features.

We used 45 machines of a 64-node Beowulf cluster to estimate the dependency model, with an average memory usage of approximately 550 MB for each machine. For the normal-form model we were able to reduce the size of the charts considerably by applying two types of restriction to the parser: first, categories can only combine if they appear together in a rule instantiation in sections 2–21 of CCGbank; and second, we apply the normal-form restrictions described in Eisner (1996). (See Clark and Curran (2004) for a description of the Eisner constraints.) The normal-form model requires only 5 machines for estimation, with an average memory usage of 730 MB for each machine.

Initially we tried the parallel version of GIS described in Clark and Curran (2003) to perform the estimation, running over the Beowulf cluster. However, we found that GIS converged extremely slowly; this is in line with other recent results in the literature applying GIS to globally optimised models such as conditional random fields, e.g. Sha and Pereira (2003). As an alternative to GIS, we have implemented a parallel version of our L-BFGS code using the Message Passing Interface (MPI) standard. L-BFGS over forests can be parallelised, using the method described in Clark and Curran (2003) to calculate the feature expectations. The L-BFGS algorithm, run to convergence on the cluster, takes 479 iterations and 2 hours for the normal-form model, and 1,550 iterations and roughly 17 hours for the dependency model.

7 Parsing Algorithm

For the normal-form model, the Viterbi algorithm is used to find the most probable derivation. For the dependency model, the highest scoring dependency structure is required. Clark and Curran (2003) outlines an algorithm for finding the most probable dependency structure, which keeps track of the highest scoring set of dependencies for each node in the chart. For a set of equivalent entries in the chart (a disjunctive node), this involves summing over all conjunctive node daughters which head subderivations leading to the same set of high scoring dependencies. In practice large numbers of such conjunctive nodes lead to very long parse times.

As an alternative to finding the most probable dependency structure, we have developed an algorithm which maximises the expected labelled recall over dependencies. Our algorithm is based on Goodman's (1996) labelled recall algorithm for the phrase-structure PARSEVAL measures.

Let L_π be the number of correct dependencies in π with respect to a gold standard dependency structure G; then the dependency structure, $\pi_{\max}$, which maximises the expected recall rate is:

$$\begin{aligned}\pi_{\max} &= \arg\max_{\pi} E(L_\pi/|G|) \qquad (10)\\ &= \arg\max_{\pi} \sum_{\pi_i} P(\pi_i|S)|\pi \cap \pi_i|\end{aligned}$$

where S is the sentence for gold standard dependency structure G and π_i ranges over the dependency structures for S. This expression can be expanded further:

$$\begin{aligned}\pi_{\max} &= \arg\max_{\pi} \sum_{\pi_i} P(\pi_i|S) \sum_{\tau\in\pi} 1 \text{ if } \tau \in \pi_i\\ &= \arg\max_{\pi} \sum_{\tau\in\pi} \sum_{\pi'|\tau\in\pi'} P(\pi'|S)\\ &= \arg\max_{\pi} \sum_{\tau\in\pi} \sum_{d\in\Delta(\pi')|\tau\in\pi'} P(d|S) \qquad (11)\end{aligned}$$

The final score for a dependency structure π is a sum of the scores for each dependency τ in π; and the score for a dependency τ is the sum of the probabilities of those derivations producing τ. This latter sum can be calculated efficiently using inside and outside scores:

$$\pi_{\max} = \arg\max_{\pi} \sum_{\tau\in\pi} \frac{1}{Z_S} \sum_{c\in C} \phi_c \psi_c \text{ if } \tau \in deps(c) \qquad (12)$$

where ϕ_c is the inside score and ψ_c is the outside score for node c (see Clark and Curran (2003)); C is the set of conjunctive nodes in the packed chart for sentence S and $deps(c)$ is the set of dependencies on conjunctive node c. The intuition behind the expected recall score is that a dependency structure scores highly if it has dependencies produced by high scoring derivations.[4]

The algorithm which finds $\pi_{\max}$ is a simple variant on the Viterbi algorithm, efficiently finding a derivation which produces the highest scoring set of dependencies.

8 Experiments

Gold standard dependency structures were derived from section 00 (for development) and section 23 (for testing) by running the parser over the derivations in CCGbank, some of which the parser could not process. In order to increase the number of test sentences, and to allow a fair comparison with other CCG parsers, extra rules were encoded in the parser (but we emphasise these were only used to obtain the section 23 test data; they were not used to parse unseen data as part of the testing). This resulted in 2,365 dependency structures for section 23 (98.5% of the full section), and 1,825 (95.5%) dependency structures for section 00.

	LP	LR	UP	UR	cat
Dep model	86.7	85.6	92.6	91.5	93.5
N-form model	86.4	86.2	92.4	92.2	93.6

Table 1: Results on development set; labelled and unlabelled precision and recall, and lexical category accuracy

Features	LP	LR	UP	UR	cat
RULES	82.6	82.0	89.7	89.1	92.4
+HEADS	83.6	83.3	90.2	90.0	92.8
+DEPS	85.5	85.3	91.6	91.3	93.5
+DISTANCE	86.4	86.2	92.4	92.2	93.6
FINAL	87.0	86.8	92.7	92.5	93.9

Table 2: Results on development set for the normal-form models

The first stage in parsing the test data is to apply the supertagger. We use the novel strategy developed in Clark and Curran (2004): first assign a small number of categories (approximately 1.4) on average to each word, and increase the number of categories if the parser fails to find an analysis. We were able to parse 98.9% of section 23 using this strategy. Clark and Curran (2004) shows that this supertagging method results in a highly efficient parser.

For the normal-form model we returned the dependency structure for the most probable derivation, applying the two types of normal-form constraints described in Section 6. For the dependency model we returned the dependency structure with the highest expected labelled recall score.

Following Clark et al. (2002), evaluation is by precision and recall over dependencies. For a labelled dependency to be correct, the first 4 elements of the dependency tuple must match exactly. For an unlabelled dependency to be correct, the heads of the functor and argument must appear together in some relation in the gold standard (in any order). The results on section 00, using the feature sets described earlier, are given in Table 1, with similar results overall for the normal-form model and the dependency model. Since experimentation is easier with the normal-form model than the dependency model, we present additional results for the normal-form model.

Table 2 gives the results for the normal-form model for various feature sets. The results show that each additional feature type increases perfor-

[4] Coordinate constructions can create multiple dependencies for a single argument slot; in this case the score for the multiple dependencies is the average of the individual scores.

	LP	LR	UP	UR	cat
Clark et al. 2002	81.9	81.8	90.1	89.9	90.3
Hockenmaier 2003	84.3	84.6	91.8	92.2	92.2
Log-linear	**86.6**	**86.3**	**92.5**	**92.1**	**93.6**
Hockenmaier(POS)	83.1	83.5	91.1	91.5	91.5
Log-linear (POS)	**84.8**	**84.5**	**91.4**	**91.0**	**92.5**

Table 3: Results on the test set

mance. Hockenmaier also found the dependencies to be very beneficial — in contrast to recent results from the lexicalised PCFG parsing literature (Gildea, 2001) — but did not gain from the use of distance measures. One of the advantages of a log-linear model is that it is easy to include additional information, such as distance, as features.

The FINAL result in Table 2 is obtained by using a larger derivation space for training, created using more categories per word from the supertagger, 2.9, and hence using charts containing more derivations. (15 machines were used to estimate this model.) More investigation is needed to find the optimal chart size for estimation, but the results show a gain in accuracy.

Table 3 gives the results of the best performing normal-form model on the test set. The results of Clark et al. (2002) and Hockenmaier (2003a) are shown for comparison. The dependency set used by Hockenmaier contains some minor differences to the set used here, but “evaluating” our test set against Hockenmaier’s gives an F-score of over 97%, showing the test sets to be very similar. The results show that our parser is performing significantly better than that of Clark et al., demonstrating the benefit of derivation features and the use of a sound statistical model.

The results given so far have all used gold standard POS tags from CCGbank. Table 3 also gives the results if automatically assigned POS tags are used in the training and testing phases, using the C&C POS tagger (Curran and Clark, 2003). The performance reduction is expected given that the supertagger relies heavily on POS tags as features.

More investigation is needed to properly compare our parser and Hockenmaier’s, since there are a number of differences in addition to the models used: Hockenmaier effectively reads a lexicalised PCFG off CCGbank, and is able to use all of the available training data; Hockenmaier does not use a supertagger, but does use a beam search.

Parsing the 2,401 sentences in section 23 takes 1.6 minutes using the normal-form model, and 10.5 minutes using the dependency model. The difference is due largely to the normal-form constraints used by the normal-form parser. Clark and Curran (2004) shows that the normal-form constraints significantly increase parsing speed and, in combination with *adaptive supertagging*, result in a highly efficient wide-coverage parser.

As a final oracle experiment we parsed the sentences in section 00 using the correct lexical categories from CCGbank. Since the parser uses only a subset of the lexical categories in CCGbank, 7% of the sentences could not be parsed; however, the labelled F-score for the parsed sentences was almost 98%. This very high score demonstrates the large amount of information in lexical categories.

9 Conclusion

A major contribution of this paper has been the development of a parsing model for CCG which uses all derivations, including non-standard derivations. Non-standard derivations are an integral part of the CCG formalism, and it is an interesting question whether efficient estimation and parsing algorithms can be defined for models which use all derivations. We have answered this question, and in doing so developed a new parsing algorithm for CCG which maximises expected recall of dependencies.

We would like to extend the dependency model, by including the local-rule dependencies which are used by the normal-form model, for example. However, one of the disadvantages of the dependency model is that the estimation process is already using a large proportion of our existing resources, and extending the feature set will further increase the execution time and memory requirement of the estimation algorithm.

We have also shown that a normal-form model performs as well as the dependency model. There are a number of advantages to the normal-form model: it requires less space and time resources for estimation and it produces a faster parser. Our normal-form parser significantly outperforms the parser of Clark et al. (2002) and produces results at least as good as the current state-of-the-art for CCG parsing. The use of adaptive supertagging and the normal-form constraints result in a very efficient wide-coverage parser. Our system demonstrates that accurate and efficient wide-coverage CCG parsing is feasible.

Future work will investigate extending the feature sets used by the log-linear models with the aim of further increasing parsing accuracy. Finally, the oracle results suggest that further experimentation with the supertagger will significantly improve parsing accuracy, efficiency and robustness.

Acknowledgements

We would like to thank Julia Hockenmaier for the use of CCGbank and helpful comments, and Mark Steedman for guidance and advice. Jason Baldridge, Frank Keller, Yuval Krymolowski and Miles Osborne provided useful feedback. This work was supported by EPSRC grant GR/M96889, and a Commonwealth scholarship and a Sydney University Travelling scholarship to the second author.

References

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, Pittsburgh, PA.

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. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, Geneva, Switzerland.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334, Philadelphia, PA.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Meeting of the ACL*, pages 184–191, Santa Cruz, CA.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Meeting of the ACL*, pages 79–86, Santa Cruz, CA.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the EMNLP Conference*, pages 167–202, Pittsburgh, PA.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the 34th Meeting of the ACL*, pages 177–183, Santa Cruz, CA.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

Julia Hockenmaier. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Meeting of the ACL*, pages 359–366, Sapporo, Japan.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York, USA.

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.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the HLT/NAACL Conference*, pages 213–220, Edmonton, Canada.

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

Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 253–263, Sozopol, Bulgaria.