Morphological analysis can improve a CCG parser for English

Matthew Honnibal, Jonathan K. Kummerfeld and James R. Curran
School of Information Technologies
University of Sydney
{mhonn, jono, james}@it.usyd.edu.au

Abstract

Because English is a low morphology language, current statistical parsers tend to ignore morphology and accept some level of redundancy. This paper investigates how costly such redundancy is for a lexicalised grammar such as CCG.

We use morphological analysis to split verb inflectional suffixes into separate tokens, so that they can receive their own lexical categories. We find that this improves accuracy when the splits are based on correct POS tags, but that errors in gold standard or automatically assigned POS tags are costly for the system. This shows that the parser can benefit from morphological analysis, so long as the analysis is correct.

1 Introduction

English is a configurational language, so grammatical functions are mostly expressed through word order and function words, rather than with inflectional morphology. Most English verbs have four forms, and none have more than five. Most of the world's languages have far richer inflectional morphology, some with millions of possible inflection combinations.

There has been much work on addressing the sparse data problems rich morphology creates, but morphology has received little attention in the English statistical parsing literature. We suggest that English morphology may prove to be an underutilised aspect of linguistic structure that can improve the performance of an English parser. English also has a rich set of resources available, so an experiment that is difficult to perform with another language may be easier to conduct in English, and a technique that makes good use of English morphology may transfer well to a morphologically rich language. under-exploited in English natural language

In this paper, we show how morphological information can improve an English statistical parser based on a lexicalised formalism, Combinatory Categorial Grammar (CCG, Steedman, 2000), using a technique suggested for Turkish (Bozsahin, 2002) and Korean (Cha et al., 2002). They describe how a morphologically rich language can be analysed efficiently with CCG by splitting off inflectional affixes as morphological tokens. This allows the affix to receive a category that performs the feature coercion. For instance, *sleeping* would ordinarily be assigned the category $S[ng]\backslash NP$: a sentence with the $[ng]$ feature requiring a leftward NP argument. We split the word into two tokens:

sleep	-ing
$S[b]\backslash NP$	$(S[ng]\backslash NP)\backslash(S[b]\backslash NP)$

The additional token creates a separate space for inflectional information, factoring it away from the argument structure information.

Even with only 5 verb forms in English, we found that accurate morphological analysis improved parser accuracy. However, the system had trouble recovering from analysis errors caused by incorrect POS tags.

We then tested how inflection categories interacted with *hat categories*, a linguistically-motivated extension to the formalism, proposed by Honnibal and Curran (2009), that introduces some sparse data problems but improves parser effiency. The parser's accuracy improved by 0.8% when gold standard POS tags were used, but not with automatic POS tags. Our method addresses problems caused by even low morphology, and future work will make the system more robust to POS tagging errors.

Coling 2008: Poster Volume, pages 445–453,
Beijing, August 2010

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG, Steedman, 2000) is a lexicalised grammar, which means that each word in the sentence is associated with a category that specifies its argument structure and the type and features of the constituent that it heads. For instance, *in* might head a PP-typed constituent with one NP-typed argument, written as PP/NP. The $/$ operator denotes an argument to the right; $\backslash$ denotes an argument to the left. For example, a transitive verb is a function from a rightward NP to and a leftward NP to a sentence, $(S\backslash NP)/NP$. The grammar consists of a few schematic rules to combine the categories:

$$\begin{array}{rcll} X/Y & Y & \Rightarrow_{>} & X \\ Y & X\backslash Y & \Rightarrow_{<} & X \\ X/Y & Y/Z & \Rightarrow_{>\mathbf{B}} & X/Z \\ Y\backslash Z & X\backslash Y & \Rightarrow_{<\mathbf{B}} & X\backslash Z \\ Y/Z & X\backslash Y & \Rightarrow_{<\mathbf{B}_{\times}} & X/Z \end{array}$$

CCGbank (Hockenmaier and Steedman, 2007) extends this grammar with a set of type-changing rules, designed to strike a better balance between sparsity in the category set and ambiguity in the grammar. We mark such productions **TC**.

In wide-coverage descriptions, categories are generally modelled as typed feature structures (Shieber, 1986), rather than atomic symbols. This allows the grammar to include head indices, and to unify under-specified features. In our notation features are annotated in square-brackets, e.g. $S[dcl]$. Head-finding indices are annotated on categories as subscripts, e.g. $(NP_y\backslash NP_y)/NP_z$. We occasionally abbreviate $S\backslash NP$ as VP, and $S[adj]\backslash NP$ as ADJ.

2.1 Statistical CCG parsing and morphology

In CCGbank, there are five features that are largely governed by the inflection of the verb:

writes/wrote	$(S[dcl]\backslash NP)/NP$
(was) written	$(S[pss]\backslash NP)/NP$
(has) written	$(S[pt]\backslash NP)/NP$
(is) writing	$(S[ng]\backslash NP)/NP$
(to) write	$(S[b]\backslash NP)/NP$

The features are necessary for satisfactory analyses. Without inflectional features, there is no way to block over-generation like *has running* or *was ran*. However, the inflectional features also create a level of redundancy if the different inflected forms are treated as individual lexical entries. The different inflected forms of a verb will all share the same set of potential argument structures, so some way of grouping the entries together is desirable.

Systems like the PET HPSG parser (Oepen et al., 2004) and the XLE LFG parser (Butt et al., 2006) use a set of lexical rules that match morphological operations with transformations on the lexical categories. For example, a lexical rule is used to ensure that an intransitive verb like *sleeping* receives the same argument structure as the base form *sleep*, but with the appropriate inflectional feature. This scheme works well for rule-based parsers, but it is less well suited for statistical parsers, as the rules propose categories but do not help the model estimate their likelihood or assign them feature weights.

Statistical parsers for lexicalised formalisms such as CCG are very sensitive to the number of categories in the lexicon and the complexity of the mapping between words and categories. The sub-task of assigning lexical categories, *supertagging* (Bangalore and Joshi, 1999), is most of the parsing task. Supertaggers mitigate sparse data problems by using a label frequency threshold to prune rare categories from the search space. Clark and Curran (2007) employ a tag dictionary that restricts the model to assigning word/category pairs seen in the training data for frequent words.

The tag dictionary causes some level of undergeneration, because not all valid word/category pairs will occur in the limited training data available. The morphological tokens we introduce help to mitigate this, by bringing together what were distinct verbs and argument structures, using lemmatisation and factoring inflection away from argument structures. The tag dictionaries for the inflectional morphemes will have very high coverage, because there are only a few inflectional categories and a few inflectional types.

3 Inflectional Categories

We implement the morphemic categories that have been discussed in the CCG literature

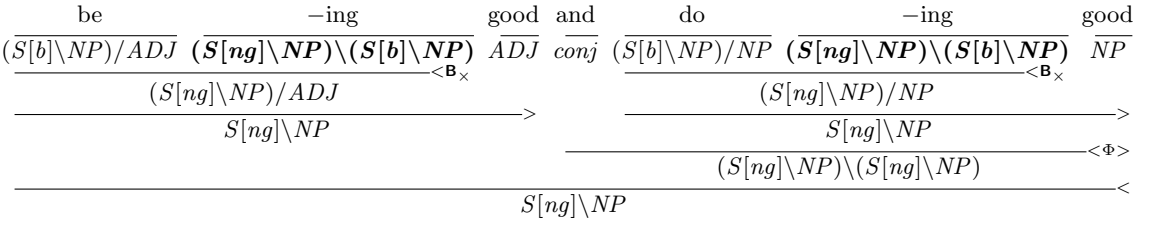

Figure 1: A single inflection category (in bold) can serve many different argument structures.

(Bozsahin, 2002; Cha et al., 2002). The inflected form is broken into two morphemes, and each is assigned a category. The category for the inflectional suffix is a function from a category with the bare-form feature $[b]$ to a category that has an inflectional feature. This prevents verbal categories from having to express their inflectional features directly. Instead, their categories only have to express their argument structure.

The CCG combinators allow multiple argument structures to share a single inflectional category. For instance, the $(S[ng]\NP)\(S[b]\NP)$ category can supply the $[ng]$ feature to all categories that have one leftward NP argument and any number of rightward arguments, via the generalised backward composition combinator. Figure 1 shows this category transforming two different argument structures, using the backward crossed composition rule ($<\mathbf{B}_\times$).

Table 1 shows the most frequent inflection categories we introduce. The majority of inflected verbs in the corpus have a subject and some number of rightward arguments, so we can almost assign one category per feature. The most frequent exceptions are participles that function as pre-nominal modifiers and verbs of speech.

Table 2 shows the inflectional token types we introduce and which features they correspond to. Our scheme largely follows the Penn Treebank tag set (Bies et al., 1995), except we avoided distinguishing past participles from past tense (*-en* vs *-ed*), because this distinction was a significant source of errors for our morphological analysis process, which relies on the part-of-speech tag.

3.1 Creating Training Data

We prepared a version of CCGbank (Hockenmaier and Steedman, 2007) with inflectional tokens. This involved the following steps:

Correcting POS tags: Our morphological analysis relies on the part-of-speech tags provided with CCGbank. We identified and corrected words whose POS tags were inconsistent with their lexical category, as discussed in Section 3.2.

Freq.	Category	Example
32.964	$(S[dcl]\NP)\(S[b]\NP)$	*He ran*
11,431	$(S[pss]\NP)\(S[b]\NP)$	*He was run down*
11,324	$(S[ng]\NP)\(S[b]\NP)$	*He was running*
4,343	$(S[pt]\NP)\(S[b]\NP)$	*He has run*
3,457	$(N/N)\(S[b]\NP)$	*the running man*
2,011	$S[dcl]\S$	*"..", he says*
1,604	$(S[dcl]\S)\(S[b]\S)$	*"..", said the boy*
169	$(S[dcl]\ADJ)\(S[b]\ADJ)$	*Here 's the deal*
55	$(S[dcl]\PP)\(S[b]\PP)$	*On it was a bee*

Table 1: The inflectional categories introduced.

Token	POS	Feat	Example
-es	VBZ	dcl	*He write -es letters*
-e	VBP	dcl	*They write -e letters*
-ed	VBD	dcl	*They write -ed letters*
-ed	VBN	pt	*They have write -ed letters*
-ed	VBN	pss	*Letters were write -ed*
-ing	VBG	ng	*They are write -ing letters*

Table 2: The inflectional token types introduced.

Lemmatising verbs and removing features: We used the morphy WordNet lemmatiser implemented in NLTK[1] to recover the lemma of the inflected verbs, identified by their POS tag (VBP, VBG, VBN or VBZ). The verb's categories were updated by switching their features to $[b]$.

Deriving inflectional categories: The generalised backward composition rules allow a functor to generalise over some sequence of argument categories, so long as they all share the same directionality. For instance, a functor $(S\NP)\(S\NP)$ could backward cross-compose into a category $((S\NP)/NP)/PP$ to its left, generalising over the two rightward arguments that were not specified by the functor's argument. It could not, however, compose into a category like $((S\NP)\NP)/PP$, because the two arguments (NP and PP) have differing direc-

[1] http://www.nltk.org

Freq.	From	To	Examples
1056	VBG	IN	*including, according, following*
379	VBN	JJ	*involved, related, concerned*
351	VBN	IN	*compared, based, given*
274	VBG	NN	*trading, spending, restructuring*
140	VBZ	NN	*is, 's, has*
102	VB	VBP	*sell, let, have*
53	VBZ	MD	*does, is, has*
45	VBG	JJ	*pending, missing, misleading*
41	VBP	MD	*do, are, have*
40	VBD	MD	*did, were, was*
334	All others		
2,815	Total		

Table 3: The most frequent POS tag conversions.

tionalities (leftward and rightward).

Without this restriction, we would only require one inflection category per feature, using inflectional categories like $S[ng]\backslash S[b]$. Instead, our inflectional categories must subcategorise for every argument except the outermost directionally consistent sequence. We discard this outermost consistent sequence, remove all features, and use the resulting category as the argument and result. We then restore the result's feature, and set the argument's feature to $[b]$.

Inserting inflectional tokens: Finally, the inflectional token is inserted after the verb, with a new node introduced to preserve binarisation.

3.2 POS tag corrections

Hockenmaier and Steedman (2007) corrected several classes of POS tag errors in the Penn Treebank when creating CCGbank. We follow Clark and Curran (2007) in using their corrected POS labels, but found that there were still some words with inconsistent POS tags and lexical categories, such as `building|NN|(S[dcl]\NP)/NP`.

In order to make our morphological analysis more consistent, we identify and correct such POS tagging errors as follows. We use two regular expressions to identify verbal lexical categories and verbal POS tags: `^\(*S\[(dcl|pss|ng|pt|b)\]` and `AUX|MD|V..` respectively. If a word has a verbal lexical category and non-verbal POS, we correct its POS tag with reference to its suffix and its category's inflectional feature. If a word has a verbal POS tag and a non-verbal lexical category, we select the POS tag that occurs most frequently with its lexical category.

The only exception are verbs functioning as nominal modifiers, such as *running* in *the running man*, which are generally POS tagged VBG but receive a lexical category of N/N. We leave these POS tagged as verbs, and instead analyse their suffixes as performing a form-function transformation that turns them from $S[b]\backslash NP$ verbs into N/N adjectives — $(N/N)\backslash(S[b]\backslash NP)$.

Table 3 lists the most common before-and-after POS tag pairs from our corrections, and the words that most frequently exemplified the pair. When compiling the table some clear errors came to light, such as the 'correction' of `is|VBZ` to `is|NN`. These errors may explain why the POS tagger's accuracy drops by 0.1% on the corrected set, and suggest that the problem of aligning POS tags and supertags is non-trivial.

In light of these errors, we experimented with an alternate strategy. Instead of correcting the POS tags, we introduced null inflectional categories that compensated for bad morphological tokenisation such as `accord|VBG|(S/S)/PP -ing|VIG|-`. The null inflectional category does not interact with the rest of the derivation, much like a punctuation symbol. This performed little better than the baseline, showing that the POS tag corrections made an important contribution, despite the problems with our technique.

3.3 Impact on CCGbank Lexicon

Verbal categories in CCGbank (Hockenmaier and Steedman, 2007) record both the valency and the inflectional morphology of the verb they are assigned to. This means $v \times i$ categories are required, where v and i are the number of distinct argument structures and inflectional features in the grammar respectively.

The inflectional tokens we propose allow inflectional morphology to be largely factored away from the argument structure, so that roughly $v + i$ verbal categories are required. A smaller category set leads to lower category ambiguity, making the assignment decision easier.

Table 4 summarises the effects of inflection categories on the lexicon extracted from CCGbank. Clark and Curran (2007) extract a set of 425 categories from the training data (Sections 02-21) that

consists of all categories that occur at least 10 times. The frequency cut off is used because the model will not have sufficient evidence to assign the other 861 categories that occur at least once, and their distribution is heavy tailed: together, they only occur 1,426 times. We refer to the frequency filtered set as the lexicon. The parser cannot assign a category outside its lexicon, so gaps in it cause under-generation.

The CCGbank lexicon includes 159 verbal categories. There are 74 distinct argument structures and 5 distinct features among these verbal categories. The grammar Clark and Curran (2007) learn therefore under-generates, because 211 of the 370 (5×74) argument structure and feature combinations are rare or unattested in the training data. For instance, there is a $(S[dcl]\backslash NP)/PP$ category, but no corresponding $(S[b]\backslash NP)/PP$, making it impossible for the grammar to generate a sentence like *I want to talk to you*, as the correct category for *talk* in this context is missing. It would be trivial to add the missing categories to the lexicon, but a statistical model would be unable to reproduce them. There are 8 occurrences of such missing categories in Section 00, the development data.

The reduction in data sparsity brought by the inflection categories causes 22 additional argument structures to cross the frequency threshold into the lexicon. A grammar induced from this corpus is thus able to generate 480 (96×5) argument structure and feature combinations, three times as many as could be generated before.

We introduce 15 inflectional categories in the corpus. The ten most frequent are shown in Table 1. The combinatory rules allow these 15 inflection categories to serve 96 argument structures, reducing the number of verbal categories in the lexicon from 159 to 89 ($74 + 15$).

The statistics at frequency 1 are less reliable, because many of the categories may be linguistically spurious: they may be artefacts caused by annotation noise in the Penn Treebank, or the conversion heuristics used by Hockenmaier and Steedman (2007).

	$\geq$	CCGbank	+Inflect
Inflection categories	10	0	15
Argument structures	10	74	96
Verb categories generated	10	159	480
All categories	10	425	375
Inflection categories	1	0	31
Argument structures	1	283	283
Verbs categories generated	1	498	1415
All categories	1	1285	1120

Table 4: Effect of inflection tokens on the category set for categories with frequency ≥ 10 and ≥ 1

3.4 Configuration of parsing experiments

We conducted two sets of parsing experiments, comparing the impact of inflectional tokens on CCGbank (Hockenmaier and Steedman, 2007) and *hat* CCGbank (Honnibal and Curran, 2009). The experiments allow us to gauge the impact of inflectional tokens on versions of CCGbank with differing numbers of verbal categories.

We used revision 1319 of the C&C parser[2] (Clark and Curran, 2007), using the best-performing configuration they describe, which used the hybrid dependency model. The most important hyper-parameters in their configuration are the β and K values, which control the workflow between the supertagger and parser. We use the Honnibal and Curran (2009) values of these parameters in our hat category experiments, described in Section 5.

Accuracy was evaluated using labelled dependency F-scores (LF). CCG dependencies are labelled by the head's lexical category and the argument slot that the dependency fills. We evaluated the baseline and inflection parsers on the unmodified dependencies, to allow direct comparison. For the inflection parsers, we pre-processed the POS-tagged input to introduce inflection tokens, and post-processed it to remove them.

We follow Clark and Curran (2007) in not evaluating accuracy over sentences for which the parser returned no analysis. The percentage of sentences analysed is described as the parser's *coverage* (C). Speed (S) figures refer to sentences parsed per second (including failures) on a dual-CPU Pentium 4 Xeon with 4GB of RAM.

[2] `http://trac.ask.it.usyd.edu.au/candc`

4 Parsing Results on CCGbank

Table 5 compares the performance of the parser on Sections 00 and 23 with and without inflection tokens. Section 00 was used for development experiments to test different approaches, and Section 23 is the test data. Similar effects were observed on both evaluation sections.

The inflection tokens had no significant impact on speed or coverage, but did improve accuracy by 0.49% F-measure when gold standard POS tags were used, compared to the baseline. However, some of the accuracy improvement can be attributed to the POS tag corrections described in Section 3.2, so the improvement from the inflection tokens alone was 0.39%.

The POS tag corrections caused a large drop in performance when automatic POS tags were used. We attribute this to the imperfections in our correction strategy. The inflection tokens improved the accuracy by 0.39%, but this was not large enough to correct for the drop in accuracy caused by the POS changes.

Another possibility is that our morphological analysis makes POS tagger errors harder to recover from. Instead of an incorrect feature value, POS tag errors can now induce poor morphological splits such as `starl|VBG -ing|VIG`. POS tagging errors are already problematic for the C&C parser, because only the highest ranked tag is forwarded to the supertagger as a feature. Our morphological analysis strategy seems to exacerbate this error propagation problem. Curran et al. (2006) showed that using a beam of POS tags as features in the supertagger and parser mitigated the loss of accuracy from POS tagging errors. Unfortunately, with our morphological analysis strategy, POS tag variations change the tokenisation of a sentence, making parsing more complicated. Perhaps the best solution would be to address the tagging errors in the treebank more thoroughly, and reform the annotation scheme to deal with particularly persistant error cases. This might improve POS tag accuracy to a level where errors are rare enough to be unproblematic.

Despite the limited morphology in English, the inflectional tokens improved the parser's accuracy when gold standard POS tags were supplied. We attribute the increase in accuracy to the more efficient word-to-category mapping caused by replacing inflected forms with lemmas, and feature-bearing verb categories with ones that only refer to the argument structure. We examined this hypothesis by performing a further experiment, to investigate how inflection tokens interact with *hat categories*, which introduce additional verbal categories that represent form-function discrepancies.

		Gold POS			Auto POS		
		LF	S	C	LF	S	C
Baseline	00	87.19	22	99.22	85.28	24	99.11
+POS	00	87.46	24	99.16	85.04	23	99.05
+Inflect	00	87.81	24	99.11	85.33	23	98.95
Baseline	23	87.69	36	99.63	85.50	36	99.58
+POS	23	87.79	36	99.63	85.06	36	99.50
+Inflect	23	88.18	36	99.58	85.42	33	99.34

Table 5: Effect of POS changes and inflection tokens on accuracy (LF), speed (S) and coverage (C) on 00 and 23.

5 Inflection Tokens and Hat Categories

Honnibal and Curran (2009) introduce an extension to the CCG formalism, *hat categories*, as an alternative way to solve the modifier category proliferation (MCP) problem. MCP is caused when a modifier is itself modified by another modifier. For instance, in the sentence *he was injured running with scissors*, *with* modifies *running*, which modifies *injured*. This produces the category $((VP\backslash VP)\backslash(VP\backslash VP))/NP$ for *with*, a rare category that is sensitive to too much of the sentence's structure.

Hockenmaier and Steedman (2007) address MCP by adding type-changing rules to CCGbank. These type-changing rules transform specific categories. They are specific to the analyses in the corpus, unlike the standard combinators, which are schematic and language universal. Honnibal and Curran's (2009) contribution is to extend the formalism to allow these type-changing rules to be lexically specified, restoring universality to the grammar — but at the cost of sparse data problems in the lexicon. Figure 2 shows how a reduced relative clause is analysed using hat categories. The hat category $(S[pss]\backslash NP)^{NP\backslash NP}$ is subject to the *unhat* rule, which unarily replaces it with its hat, $NP\backslash NP$, allowing it to function as a modifier.

Hat categories have a practical advantage for a parser that uses a supertagging phase (Bangalore

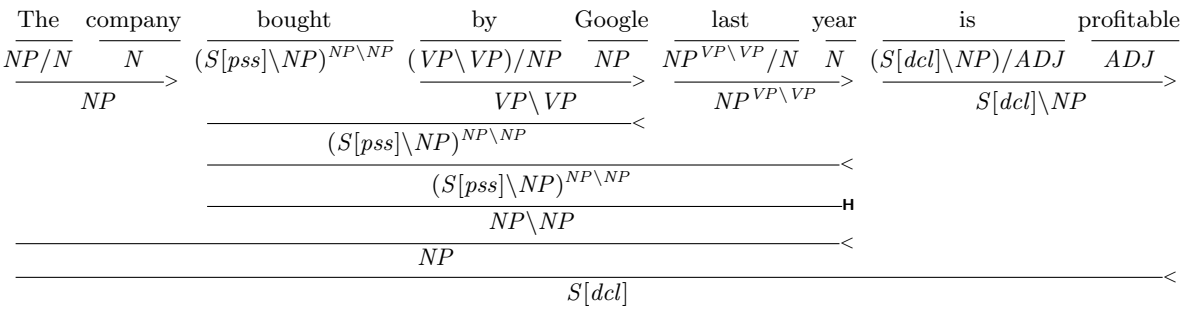

Figure 2: CCG derivation showing hat categories and the unhat rule.

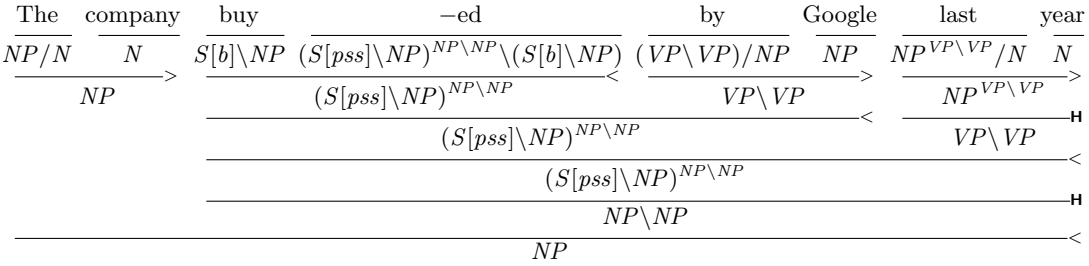

Figure 3: CCG derivation showing how inflectional tokens interact with hat categories.

and Joshi, 1999), such as the C&C system (Clark and Curran, 2007). By replacing type-changing rules with additional lexical categories, more of the work is shifted to the supertagger. The supertagging phase is much more efficient than the chart parsing stage, so redistribution of labour makes the parser considerably faster.

Honnibal and Curran (2009) found that the parser was 37% faster on the test set, at a cost of 0.5% accuracy. They attribute the drop in accuracy to sparse data problems for the supertagger, due to the increase in the number of lexical categories. We hypothesised that inflectional categories could address this problem, as the two analyses interact well.

5.1 Analyses with inflectional hat categories

Using hat categories to lexicalise type-changing rules offers attractive formal properties, and some practical advantages. However, it also misses some generalisations. A type-changing operation such as $S[ng]\backslash NP \rightarrow NP\backslash NP$ must be available to any VP. If we encounter a new word, *The company is blagging its employees*, we can generalise to the reduced relative form, *She works for that company blagging its employees* with no additional information.

This property could be preserved with some form of lexical rule, but a novel word-category pair is difficult for a statistical model to assign. Inflection tokens offer an attractive solution to this problem, as shown in Figure 3. Assigning the hat category to the suffix makes it available to any verb the suffix follows — it is just another function the inflectional suffix can perform. This generality also makes it much easier to learn, because it does not matter whether the training data happens to contain examples of a given verb perfoming that grammatical function.

We prepared a version of the Honnibal and Curran (2009) hat CCGbank, moving hats on to inflectional categories wherever possible. The hat CCGbank's lexicon contained 105 hat categories, of which 77 were assigned to inflected verbs. We introduced 33 inflection hat categories in their place, reducing the number of hat categories by 27.9%. Fewer hat categories were required because different argument structures could be served by the same inflection category. For instance, the $(S[ng]\backslash NP)^{NP\backslash NP}$ and $(S[ng]\backslash NP)^{NP\backslash NP}/NP$ categories were both replaced by the $(S[ng]\backslash NP)^{NP\backslash NP}\backslash(S[b]\backslash NP)$ category. Table 6 lists the most frequent inflection hat categories we introduce.

Freq.	Category
3332	$(S[pss]\backslash NP)^{NP\backslash NP}\backslash(S[b]\backslash NP)$
1518	$(S[ng]\backslash NP)^{NP\backslash NP}\backslash(S[b]\backslash NP)$
1231	$(S[ng]\backslash NP)^{(S\backslash NP)\backslash(S\backslash NP)}\backslash(S[b]\backslash NP)$
360	$((S[dcl]\backslash NP)/NP)^{NP\backslash NP}\backslash((S[b]\backslash NP)/NP)$
316	$(S[ng]\backslash NP)^{NP}\backslash(S[b]\backslash NP)$
234	$((S[dcl]\backslash NP)/S)^{S/S}\backslash((S[b]\backslash NP)/S)$
209	$(S[ng]\backslash NP)^{S/S}\backslash(S[b]\backslash NP)$
162	$(S[dcl]^{NP\backslash NP}\backslash NP)\backslash(S[b]\backslash NP)$
157	$((S[dcl]\backslash NP)/S)^{VP/VP}\backslash((S[b]\backslash NP)/S)$
128	$(S[pss]\backslash NP)^{S/S}\backslash(S[b]\backslash NP)$

Table 6: The most frequent inflection hat categories.

5.2 Parsing results

Table 7 shows the hat parser's performance with and without inflectional categories. We used the values for the β and K hyper-parameters described by Honnibal and Curran (2009). These hyper-parameters were tuned on Section 00, and some over-fitting seems apparent. We also followed their dependency conversion procedure, to allow evaluation over the original CCGbank dependencies and thus direct comparison with Table 5. We also merged the parser changes they described into the development version of the C&C parser we are using, for parse speed comparison.

Interestingly, incorporating the hat changes into the current version has increased the advantage of the hat categories. Honnibal and Curran report a 37% improvement in speed for the hybrid model (which we are using) on Section 23, using gold standard POS tags. With our version of the parser, the improvement is 86% (36 vs. 67 sentences parsed per second).

With gold standard POS tags, the inflection tokens improved the hat parser's accuracy by 0.8%, but decreased its speed by 24%. We attribute the decrease in speed to the increase in sentence length coupled with the new uncertainty on the inflectional tokens. Coverage increased slightly with gold standard POS tags, but decreased with automatic POS tags. We attribute this to the fact that POS tagging errors lead to morphological analysis errors.

The accuracy improvement on the hat corpus was more robust to POS tagging errors than the CCGbank results, however. This may be because POS tagging errors are already quite problematic for the hat category parser. POS tag features are more important for the supertagger than the parser, and the supertagger performs more of the work for the hat parser.

		Gold POS			Auto POS		
		LF	S	C	LF	S	C
Hat baseline	00	87.08	32	99.53	84.67	34	99.32
Hat inflect	00	87.85	37	99.63	84.99	30	98.95
Hat baseline	23	87.26	67	99.50	84.93	53	99.58
Hat inflect	23	88.06	54	99.63	85.25	43	99.38

Table 7: Effect of inflection tokens on accuracy (LF), speed (S) and coverage (C) on Sections 00 and 23.

6 Conclusion

Lexicalised formalisms like CCG (Steedman, 2000) and HPSG (Pollard and Sag, 1994) have led to high-performance statistical parsers of English, such as the C&C CCG parser (Clark and Curran, 2007) and the ENJU HPSG (Miyao and Tsuji, 2008) parser. The performance of these parsers can be partially attributed to their theoretical foundations. This is particularly true of the C&C parser, which exploits CCG's lexicalisation to divide the parsing task between two integrated models (Clark and Curran, 2004).

We have followed this formalism-driven approach by exploiting morphology for English syntactic parsing, using a strategy designed for morphologically rich languages. Combining our technique with hat categories leads to a 20% improvement in efficiency, with a 0.25% loss of accuracy. If the POS tag error problem were addressed, the two strategies combined would improve efficiency by 50%, and improve accuracy by 0.37%. These results illustrate that linguistically motivated solutions can produce substantial practical advantages for language technologies.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback, and the members of the CCG-technicians mailing list for discussion about some of our analyses. Matthew Honnibal was supported by Australian Research Council (ARC) Discovery Grant DP0665973. James Curran was supported by ARC Discovery grant DP1097291 and the Capital Markets Cooperative Research Centre.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, MS-CIS-95-06, University of Pennsylvania, Philadelphia, PA, USA.

Cem Bozsahin. 2002. The combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186.

Miriam Butt, Mary Dalrymple, and Tracy H. King, editors. 2006. CSLI Publications, Stanford, CA.

Jeongwon Cha, Geunbae Lee, and Jonghyeok Lee. 2002. Korean Combinatory Categorial Grammar and statistical parsing. *Computers and the Humanities*, 36(4):431–453.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of 20th International Conference on Computational Linguistics*, pages 282–288. Geneva, Switzerland.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 697–704. Sydney, Austrailia.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal and James R. Curran. 2009. Fully lexicalising CCGbank with hat categories. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1212–1221. Singapore.

Yusuke Miyao and Jun'ichi Tsuji. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Stepan Oepen, Daniel Flickenger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. a rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2(4):575–596.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. CSLI Publications, Stanford, CA.

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