A More Precise Analysis of Punctuation for Broad-Coverage Surface Realization with CCG

Michael White and **Rajakrishnan Rajkumar**
Department of Linguistics
The Ohio State University
Columbus, OH, USA
{mwhite,raja}@ling.osu.edu

Abstract

This paper describes a more precise analysis of punctuation for a bi-directional, broad coverage English grammar extracted from the CCGbank (Hockenmaier and Steedman, 2007). We discuss various approaches which have been proposed in the literature to constrain overgeneration with punctuation, and illustrate how aspects of Briscoe's (1994) influential approach, which relies on syntactic features to constrain the appearance of balanced and unbalanced commas and dashes to appropriate sentential contexts, is unattractive for CCG. As an interim solution to constrain overgeneration, we propose a rule-based filter which bars illicit sequences of punctuation and cases of improperly unbalanced apposition. Using the OpenCCG toolkit, we demonstrate that our punctuation-augmented grammar yields substantial increases in surface realization coverage and quality, helping to achieve state-of-the-art BLEU scores.

1 Introduction

In his pioneering monograph, Nunberg (1990) argues that punctuation is a systematic module of the grammar of written text and is governed by principles and constraints like other sub-systems such as syntax or phonology. Since then, others including Briscoe (1994) and Doran (1998) have explored ways of including rules and representations for punctuation marks in broad coverage grammars. In computational systems, punctuation provides disambiguation cues which can help parsers arrive at the correct parse. From a natural language generation standpoint, text without punctuation can be difficult to comprehend, or even misleading.

In this paper, we describe a more precise analysis of punctuation for a bi-directional, broad coverage English grammar extracted from the CCGbank (Hockenmaier and Steedman, 2007). In contrast to previous work, which has been primarily oriented towards parsing, our goal has been to develop an analysis of punctuation that is well suited for both parsing and surface realization. In addition, while Briscoe and Doran have simply included punctuation rules in their manually written grammars, our approach has been to revise the CCGbank itself with punctuation categories and more precise linguistic analyses, and then to extract a grammar from the enhanced corpus.

In developing our analysis, we illustrate how aspects of Briscoe's (1994) approach, which relies on syntactic features to constrain the appearance of balanced and unbalanced commas and dashes to appropriate sentential contexts, is unattractive for CCG, with its more flexible handling of word order. Consequently, as an interim solution, we have chosen to identify and filter undesirable configurations when scoring alternative realizations. We also point to other ways in which punctuation constraints could be incorporated into the grammar, for exploration in future work.

Using the OpenCCG toolkit, we demonstrate that our punctuation-enhanced grammar yields substantial increases in surface realization quality, helping to achieve state-of-the-art BLEU scores. We use non-blind testing to evaluate the efficacy of the grammar, and blind-testing to evaluate its performance on unseen data. The baseline models

Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks, pages 17–24
Manchester, August 2008

are (1) a grammar which has lexicalized punctuation categories only for conjunction and apposition, and (2) one which has punctuation categories corresponding to the existing treatment of punctuation in the corpus. Non-blind testing results shown a nearly 9-point increase in BLEU scores compared to the best baseline model using oracle n-grams, as well as a 40% increase in exact matches. Blind testing results show a more than 5.5-point increase in BLEU scores, contributing to an all-sentences score of 0.7323 on Section 23 with over 96% coverage.

2 Background

CCG (Steedman, 2000) is a unification-based categorial grammar formalism which is defined almost entirely in terms of lexical entries that encode sub-categorization information as well as syntactic feature information (e.g. number and agreement). Complementing function application as the standard means of combining a head with its argument, type-raising and composition support transparent analyses for a wide range of phenomena, including right-node raising and long distance dependencies. Semantic composition happens in parallel with syntactic composition, which makes it attractive for generation.

OpenCCG is a parsing/generation library which works by combining lexical categories for words using CCG rules and multi-modal extensions on rules (Baldridge, 2002) to produce derivations. Surface realization is the process by which logical forms are transduced to strings. OpenCCG uses a hybrid symbolic-statistical chart realizer (White, 2006) which takes logical forms as input and produces sentences by using CCG combinators to combine signs. Alternative realizations are ranked using integrated n-gram scoring.

To illustrate the input to OpenCCG, consider the semantic dependency graph in Figure 1. In the graph, each node has a lexical predication (e.g. **make.03**) and a set of semantic features (e.g. ⟨NUM⟩sg); nodes are connected via dependency relations (e.g. ⟨ARG0⟩). Internally, such graphs are represented using Hybrid Logic Dependency Semantics (HLDS), a dependency-based approach to representing linguistic meaning (Baldridge and Kruijff, 2002). In HLDS, each semantic head (corresponding to a node in the graph) is associated with a nominal that identifies its discourse referent, and relations between heads and their dependents

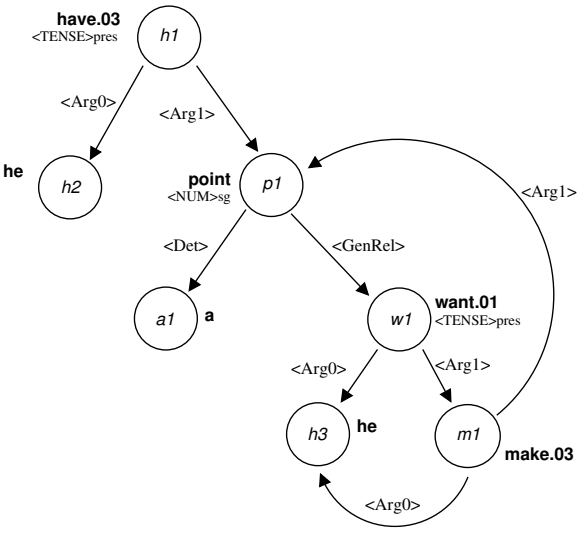

Figure 1: Semantic dependency graph from the CCGbank for *He has a point he wants to make [. . .]*

are modeled as modal relations.

3 The need for an OpenCCG analysis of punctuation

The linguistic analysis aims to make a broad coverage OpenCCG grammar extracted from the CCGbank (White et al., 2007) more precise by adding lexicalized punctuation categories to deal with constructions involving punctuation. The original CCGbank corpus does not have lexical categories for punctuation; instead, punctuation marks carry categories derived from their part of speech tags and form part of a binary rule. It is assumed that there are no dependencies between words and punctuation marks and that the result of punctuation rules is the same as the non-punctuation category. OpenCCG does not support non-combinatory binary rules, as they can be replaced by equivalent lexicalized categories with application-only slashes. For example, a binary rule of the form , $\mathsf{s} \Rightarrow \mathsf{s}$ can be replaced by the equivalent category $\mathsf{s}_{\langle 1 \rangle}/_{\star}\mathsf{s}_{\langle 1 \rangle}$ for the comma. In fact, this would work reasonably well for parsing, but is inadequate for generation. To illustrate, consider (1):

(1) Despite recent declines in yields, investors continue to pour cash into money funds. (wsj_0004.10)

A comma category like the one shown above would end up overgenerating, as sentences and

sentential complements would be generated with a comma preceding them. Also, the result of the above function application rule could act as its own argument, producing a string of commas. More generally, binary rules miss out on many linguistic generalizations, such as the presence of mandatory balancing marks in sentence-medial comma or dash adjuncts.

The literature discusses various means to address the issue of overgeneration: absorption rules (Nunberg, 1990), syntactic features (Doran, 1998) and (Briscoe, 1994) and semantic features (White, 2006). Section 5 explains these approaches in detail, and considers a possible system of syntactic features for a multi-modal CCG grammar implementation. We show how such a system is inadequate to constrain all possible cases of overgeneration, motivating our decision to employ semantic features in our bi-directional grammar.

4 Integrating an analysis of punctuation into the grammar

As our starting point, we used an XML representation of an enhanced version of the CCGbank with Propbank roles projected onto it (Boxwell and White, 2008). Contexts and constructions in which punctuation marks occur were isolated and the corpus was then restructured by inserting new categories and modified derivations using XSL transforms. In many cases this also involved modifying the gold standard derivations substantially and adding semantic representations to syntactic categories using logical form templates. Currently, the algorithm succeeds in creating logical forms for 98.01% of the sentences in the development section (Sect. 00) of the converted CCGbank, and 96.46% of the sentences in the test section (Sect. 23). Of these, 92.10% of the development LFs are semantic dependency graphs with a single root, while 92.12% of the test LFs have a single root. The remaining cases, with multiple roots, are missing one or more dependencies required to form a fully connected graph. These missing dependencies usually reflect inadequacies in the current logical form templates. In Section 00, 89 punctuation categories were created (66 commas, 14 dashes and 3 each for the rest) out of 54 classes of binary rules (37 comma, 8 dash, 3 apiece of colon, parenthesis and dots). Three high frequency comma categories are explained below.

4.1 Sentential Adjuncts

The comma in example (1) has been analysed as selecting a sentential modifier to its left, *Despite recent declines in yields*, to result in a sentential modifier which then selects the rest of the sentence. This results in the following lexical category and semantics for the comma category:

(2) , $\vdash \mathsf{s}_{\langle 1\rangle ind=X1,mod=M}/\mathsf{s}_{\langle 1\rangle}\backslash_{\star}(\mathsf{s}_{\langle 1\rangle}/\mathsf{s}_{\langle 1\rangle})$
: $@_M(\langle\text{EMPH-INTRO}\rangle+)$

Syntactic categories and their semantics are linked by index variables in the feature structures of categories. Index variables for semantic heads (e.g. $X1$) are conventionally named X plus the number of the feature structure. To support modifier modifiers, as in (2), semantic heads of modifiers are also made available through a modifier index feature, with a variable conventionally named M.[1] Here, the effect of combining the comma with the phrase headed by *despite* is to add the $\langle$EMPH-INTRO$\rangle+$ feature to the *despite*-phrase's semantics. Following (Bayraktar et al., 1998), this feature indicates that the comma has the discourse function of emphasizing an introductory clause or phrase. During realization, the feature triggers the look-up of the category in (2), and prevents the re-application of the category to its own output (as the feature should only be realized once).

The category in (2) illustrates our approach, which is to assign to every punctuation mark (other than balancing marks) a category whose LF includes a feature or relation which represents its discourse semantic function in broad-brush terms such as emphasis, elaboration and apposition.

4.2 Verbs of reported speech

In (3), the comma which follows *Nevertheless* and sets off the phrase headed by *said* has the category in (4):

(3) Nevertheless, said Brenda Malizia Negus, editor of Money Fund Report, yields may blip up again before they blip down because of recent rises in short-term interest rates. (wsj_0004.8)

(4) , $\vdash \mathsf{s}_{\langle 2\rangle}/\mathsf{s}_{\langle 2\rangle}/_{\star}\mathsf{punct}[,]/_{\star}(\mathsf{s}_{\langle 1\rangle dcl}\backslash\mathsf{s}_{\langle 2\rangle dcl})$
: $@_{X2}(\langle\text{ELABREL}\rangle \wedge X1)$

[1] A limited form of default unification is used in the implementation to keep multiple modifiers from conflicting. As the names of index variables are entirely predictable, they are suppressed in the remainder of the paper.

In the genre of newswire text, this construction occurs frequently with verbs of reported speech. The CCGbank derivation of (3) assigns the category $\mathsf{s}_{\langle 1\rangle dcl}\backslash\mathsf{s}_{\langle 2\rangle dcl}$ to the phrase headed by *said*, the same category that is used when the phrase follows the missing sentential complement. The comma category in (4) selects for this category and a balancing comma and then converts it to a pre-sentential modifier, $\mathsf{s}_{\langle 2\rangle}/\mathsf{s}_{\langle 2\rangle}$. Semantically, an elaboration relation is added between the main clause and the reported speech phrase.

Category (4) overgenerates to some extent in that it will allow a comma at the beginning of the sentence. To prevent this, an alternative would be to make the comma explicitly select for lexical material to its left (in this case for the category of *Nevertheless*). Another possibility would be to follow Doran (1998) in analyzing the above construction by using the verb itself to select for the comma. However, since our method involves changing the gold standard derivations, and since making the verb select extra commas or having the comma select leftward material would entail substantial further changes to the derivations, we have opted to go with (4), balancing adequacy and convenience.

4.3 NP appositives

Neither the Penn Tree Bank nor the CCGbank distinguishes between NP appositives and NP conjunctions. We wrote a set of simple heuristic rules to enforce this distinction, which is vital to generation. Appositives can occur sentence medially or finally. The conventions of writing mandate that sentence medial appositives should be balanced—i.e., the appositive NP should be surrounded by commas or dashes on both sides—while sentence final appositives should be unbalanced—i.e., they should only have one preceding comma or dash. The categories and semantics for unbalanced and balanced appositive commas are, respectively:

(5) a. $, \vdash \mathsf{np}_{\langle 1\rangle}\backslash\mathsf{np}_{\langle 1\rangle}/_{\star}\mathsf{np}_{\langle 3\rangle}$
$: @_{X1}(\langle \textsc{AppositiveRel}\rangle \wedge X3)$

b. $, \vdash \mathsf{np}_{\langle 1\rangle}\backslash\mathsf{np}_{\langle 1\rangle}/_{\star}\mathsf{punct}[,]/_{\star}\mathsf{np}_{\langle 3\rangle}$
$: @_{X1}(\langle \textsc{AppositiveRel}\rangle \wedge X3)$

Here, the unbalanced appositive has a category where the comma selects as argument the appositive NP and converts it to a nominal modifier. For balanced appositives, the comma selects the appositive NP and the balancing comma to form a nominal modifier (examples are given in the next section).

5 Constraining overgeneration in bi-directional grammars

A complex issue that arises in the design of bi-directional grammars is ensuring the proper presentation of punctuation. Among other things, this involves the task of ensuring the correct realization of commas introducing noun phrase appositives—in our case, choosing when to use (5a) vs. (5b). In this section, we consider and ultimately reject a solution that follows Briscoe (1994) in using syntactic features. As an alternative, interim solution, we then describe a rule-based filter which bars illicit punctuation sequences and improperly unbalanced apposition. The paradigm below helps illustrate the issues:

(6) John, CEO of ABC, loves Mary.

(7) * John, CEO of ABC loves Mary.

(8) Mary loves John, CEO of ABC.

(9) * Mary loves John, CEO of ABC,.

(10) Mary loves John, CEO of ABC, madly.

(11) * Mary loves John, CEO of ABC madly.

5.1 Absorption vs. syntactic features

Nunberg (1990) argues that text adjuncts introduced by punctuation marks have an underlying representation where these adjuncts have marks on either side. They attain their surface form when a set of presentation rules are applied. This approach ensures that all sentence medial cases like (6) and (10) above are generated correctly, while unacceptable examples (7) and (11) would not be generated at all. Example (8) would at first be generated as (9): to deal with such sentences, where two points happen to coincide, Nunberg posits an implicit point which is absorbed by the adjacent point. Absorption occurs according to the "strength" of the two points. Strength is determined according to the Point Absorption Hierarchy, which ranks commas lower than dashes, semi-colons, colons and periods. As White (1995) observes, from a generation-only perspective, it makes sense to generate text adjuncts which are always balanced and post-process the output to delete lower ranked points, as absorption uses relatively simple rules that operate independently of

the hierarchy of the constituents. However, using this approach for parsing would involve a preprocessing step which inserts commas into possible edges of possible constituents, as described in (Forst and Kaplan, 2006). To avoid this considerable complication, Briscoe (1994) has argued for developing declarative approaches involving syntactic features, with no deletions or insertions of punctuation marks.

5.2 Features for punctuation in CCG?

Unfortunately, the feature-based approach appears to be inadequate for dealing with the class of examples presented above in CCG. This approach involves the incorporation of syntactic features for punctuation into atomic categories so that certain combinations are blocked. To ensure proper appositive balancing sentence finally, the rightmost element in the sentence should transmit a relevant feature to the clause level, which the sentence-final period can then check for the presence of right-edge punctuation. Possible categories for a transitive verb and the full stop appear below:

(12) $\textit{loves} \vdash \mathsf{s}_{\langle 1\rangle bal=BAL,end=PE} \backslash \mathsf{np}_{\langle 2\rangle bal=+} / \mathsf{np}_{\langle 3\rangle bal=BAL,end=PE}$

(13) $. \vdash \mathsf{sent} \backslash_{\star} \mathsf{s}_{end=nil}$

Here the feature variables *BAL* and *PE* of the rightmost argument of the verb would unify with the corresponding result category feature values to realize the main clauses of (8) and (9) with the following feature values:

(14) Mary loves John, CEO of ABC $\vdash$ $\mathsf{s}_{\langle 1\rangle bal=-,end=nil}$

(15) Mary loves John, CEO of ABC, $\vdash$ $\mathsf{s}_{\langle 1\rangle bal=+,end=comma}$

Thus, in (15), the sentence-final period would not combine with $\mathsf{s}_{\langle 1\rangle bal=+,end=comma}$ and the derivation would be blocked.[2]

5.2.1 Issue: Extraction cases

The solution sketched above is not adequate to deal with extraction involving ditransitive verbs in cases like (16) and (17):

(16) Mary loves a book that John gave Bill, his brother.

(17) * Mary loves a book that John gave Bill, his brother,.

As Figure 2 shows, an unacceptable case like (17) is not blocked. Even when the sentence final NP is balanced, the *end=comma* value is not propagated to the root level. This is because the *end* feature for the relative clause should depend on the first (indirect) object of *gave*, rather than the second (direct) object as in a full ditransitive clause. A possible solution would be to introduce more features which record the presence of punctuation in the leftward and rightward arguments of complex categories; this would be rather baroque, however.

5.2.2 Issue: Crossing composition

Another issue is how crossing composition, used with adverbs in heavy NP shift contructions, interacts with appositives, as in the following examples:

(18) Mary loves madly John, CEO of ABC.

(19) * Mary loves madly John, CEO of ABC,.

For examples (10) and (11), which do not involve crossing composition, the category for the adverb should be the one in (20):

(20) $\textit{madly} \vdash \mathsf{s}_{\langle 1\rangle end=nil} \backslash \mathsf{np}_{\langle 2\rangle} \backslash (\mathsf{s}_{\langle 1\rangle bal=+} \backslash \mathsf{np}_{\langle 2\rangle})$

Here the *bal=+* feature on the argument of the adverb *madly* ensures that the direct object of the verb is balanced, as in (10); otherwise, the derivation fails, as in (11). Irrespective of the value of the *end* feature of the argument, the result of the adverb has the feature *end=nil* as the post-modifier is lexical material which occurs after the VP. With crossing composition, however, category (20) would licence an erroneous derivation for example (19), as the *end=nil* feature on the result of the adverb category would prevent the percolation of the *end* feature at the edge of the phrase to the clausal root, as Figure 3 shows.

To block such derivations, one might consider giving the adverb another category for use with crossing composition:

(21) $\textit{madly} \vdash \mathsf{s}_{\langle 1\rangle} \backslash \mathsf{np}_{\langle 2\rangle} \backslash_{\times} (\mathsf{s}_{\langle 1\rangle} \backslash \mathsf{np}_{\langle 2\rangle})$

The use of the non-associative, permutative modality $\times$ on the main slash allows the crossing composition rule to be applied, and feature inheritance

[2] It is worth noting than an n-gram scorer would highly disprefer example (9), as a comma period sequence would not be attested in the training data. However, an n-gram model cannot be relied upon to eliminate examples like (11), which would likely be favored as they are shorter than their balanced counterparts.

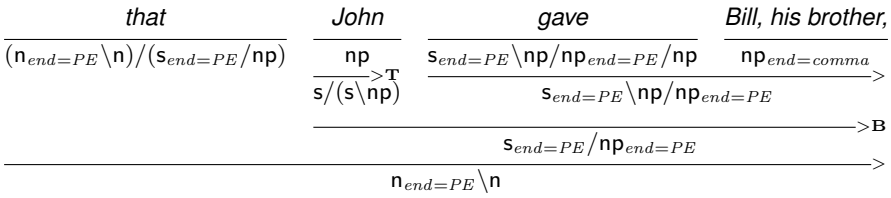

Figure 2: Object extraction

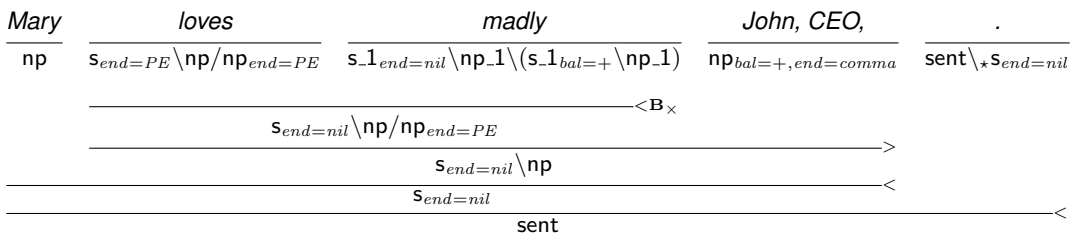

Figure 3: Crossing composition

ensures that the *end* feature from the verb *loves* is also copied over. Thus, in example (19), the punctuation at the edge of the phrase would be percolated to the clausal root, where the sentence-final period would block the derivation. However, in the slash modality inheritance hierarchy proposed by Baldridge (2002), the $\times$ modality inherits the properties of function application. Consequently, this category could also lead to the erroneous derivation of example (11). In such a derivation, category (21) will not require the direct object to have a balanced appositive; meanwhile, the *end=nil* feature on the direct object will propagate to the clausal root, where it will happily combine with the category for the full stop. Finally, having two distinct categories for the adverb would offset the advantage of multi-modal categorial grammar in dealing with word order variation, where it is possible to use one category in situations where otherwise several categories would be required.

5.3 A rule-based filter to constrain overgeneration

For the reasons discussed in the preceding section, we decided not to use syntactic features to constrain overgeneration. Instead, we have employed semantic features in the logical form together with a rule-based filter, as an interim solution. During realization, the generated output is examined and fragments where two marks appear in a row are eliminated. Additionally, to handle improperly unbalanced punctuation, we modified the result categories of unbalanced appositive commas and dashes to include a feature marking unbalanced punctuation, as follows:

(22) $\quad , \vdash \mathsf{np}_{\langle 1 \rangle unbal=comma} \backslash_{\star} \mathsf{np}_{\langle 1 \rangle} /_{\star} \mathsf{np}_{\langle 2 \rangle}$

Then, during realization, a filter on derivations looks for categories such as $\mathsf{np}_{unbal=comma}$, and checks to make sure this NP is followed by a another punctuation mark in the string. We report on the effects of the filter in our results section.

6 Evaluation

We extracted a grammar from the restructured corpus and created testbeds of logical forms under the following conditions:

1. Baseline 1: A CCGbank version which has no lexicalized categories corresponding to any of the punctuation marks except sentence final marks and commas which conjoin elements or introduce NP appositives. Conjunction and apposition are frequent in the corpus and if excluded, logical forms for many sentences are not produced, weakening the baseline considerably.

2. Baseline 2: A CCGbank version where all punctuation marks (except conjunction/apposition commas and sentence-final marks, which have proper categories) have lexicalized MMCCG categories with no semantics, corresponding to binary rules in the original CCGbank.

3. The CCGbank augmented with punctuation categories.

Testing was done under four conditions:

1. Non-blind testing with oracle n-gram scoring. This condition tests the grammar most directly, as it avoids the issue of lexical smoothing and keeps the combinatorial search manageable. A grammar extracted from the development section (Section 00) of the CCGbank was applied to the LF testbed of that section, using oracle n-gram scoring (along with FLMs, see next) to generate the sentences back. For each logical form, the generated output sentence was compared with the actual gold standard sentence corresponding to that logical form.

2. Blind testing with factored language models (FLM) and lexical smoothing, following (White et al., 2007). Blind testing naturally provides a more realistic test of performance on unseen data. Here logical forms of Sections 00 and 23 were created using grammars of those sections respectively and then a grammar was extracted from the standard training sections (02-21). This grammar was used to generate from the LFs of the development and test sections; for space reasons, we only report the results on the test section.

3. Blind testing with hypertagging. Hypertagging (Espinosa et al., 2008) is supertagging for surface realization; it improves realizer speed and coverage with large grammars by predicting lexical category assignments with a maximum entropy model.

4. The punctuation-enhanced grammars were tested in the three conditions above with and without the balanced punctuation filter.

7 Results

Non-blind testing results in Table 1 indicate that both exact match figures as well BLEU scores increase substantially in comparison to the baselines when a punctuation augmented grammar is used. The difference is especially notable when oracle n-gram scoring is used. The punctuation filter improves performance as exact matches increase by 1.66% and BLEU scores also show a slight increase. Complete realizations are slightly worse for the augmented grammar than Baseline 1, but the coverage of the baseline grammar is lower.

Table 1: Non-blind testing on Section 00 (Grammar coverage: Baseline 1, 95.8%; Baseline 2, 95.03%; Punct grammar, 98.0%)

N-grams	Grammar	Exact	Complete	BLEU
Oracle	Baseline 1	35.8%	86.2%	0.8613
	Baseline 2	39.10%	53.58%	0.8053
	Punct	75.9%	85.3%	0.9503
FLM w/o filter	Baseline 1	17.7%	83.0%	0.7293
	Baseline 2	5.72%	4.18%	0.4470
	Punct	29.7%	80.6%	0.7984
FLM w/ filt.	Punct	31.3%	80.6%	0.8062

Table 2: Blind testing on Section 23 with FLM (Grammar coverage: Baseline 1, 94.8%; Baseline 2, 95.06%; Punct grammar, 96.5%)

Hyp., Filt.	Grammar	Exact	Complete	BLEU
no, w/o	Baseline 1	11.1%	46.4%	0.6297
	Baseline 2	2.97%	3.97%	0.3104
	Punct	18.0%	43.2%	0.6815
no, w/	Punct	19.3%	43.3%	0.6868
yes, w/o	Punct	20.4%	61.5%	0.7270
yes, w/	Punct	21.6%	61.5%	0.7323

Blind testing results shown in Table 2 also demonstrate that the augmented grammar does better than the baseline in terms of BLEU scores and exact matches, with the hypertagger further boosting BLEU scores and the number of complete realizations. The use of the filter yields a further 1.2–1.3% increase in exact match figures as well as a half a BLEU point improvement; a planned collection of human judgments may reveal that these improvements are more meaningful than the scores would indicate.

Baseline 2, which models all punctuation, performs very badly with FLM scoring though it does better than the minimal punctuation Baseline 1 with oracle scoring. The main reason for this is that, without any semantic or syntactic features to constrain punctuation categories, they tend to reapply to their own output, clogging up the chart. This results in a low number of complete realizations as well as exact matches.

While direct comparisons cannot really be made across grammar frameworks, as inputs vary in their semantic depth and specificity, we observe that our all-sentences BLEU score of 0.7323 exceeds that of Hogan et al. (2007), who report a top score of 0.6882 including special treatment of multi-word units (though their coverage is near 100%). Nakanishi et al. (2005) and Langkilde-

Geary (2002) report scores several points higher, though the former is limited to sentences of length 20 or less, and the latter's coverage is much lower.

8 Conclusion

We have shown that incorporating a more precise analysis of punctuation into a broad-coverage reversible grammar extracted from the CCGbank yields substantial increases in the number of exact matches and BLEU scores when performing surface realization with OpenCCG, contributing to state-of-the-art results. Our discussion has also highlighted the inadequacy of using syntactic features to control punctuation placement in CCG, leading us to develop a filter to ensure appropriately balanced commas and dashes. In future work, we plan to investigate a more satisfactory grammatical treatment involving constraints in independent orthographic derivations, perhaps along the lines of the autonomous prosodic derivations which Steedman and Prevost (1994) discuss. An evaluation of parsing side performance is also planned.

Acknowledgments

We thank the anonymous reviewers, Detmar Meurers and the Clippers and Synners groups at OSU for helpful comments and discussion.

References

Baldridge, Jason and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.

Baldridge, Jason. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Bayraktar, Murat, Bilge Say, and Varol Akman. 1998. An Analysis of English Punctuation: The Special Case of Comma. *International Journal of Corpus Linguistics*, 3(1):33–58.

Boxwell, Stephen and Michael White. 2008. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*. To appear.

Briscoe, Ted. 1994. Parsing (with) punctuation. Technical report, Xerox, Grenoble, France.

Doran, Christine. 1998. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. Ph.D. thesis, University of Pennsylvania.

Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proc. ACL-08:HLT*. To appear.

Forst, Martin and Ronald M. Kaplan. 2006. The importance of precise tokenizing for deep grammars. In *Proc. LREC-06*.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Hogan, Deirdre, Conor Cafferkey, Aoife Cahill, and Josef van Genabith. 2007. Exploiting multi-word units in history-based probabilistic generation. In *Proc. EMNLP-CoNLL-07*.

Langkilde-Geary, Irene. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. INLG-02*.

Nakanishi, Hiroko, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proc. IWPT-05*.

Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation*. CSLI Publications, Stanford, CA.

Steedman, Mark and S. Prevost. 1994. Specifying intonation from context for speech synthesis. *Speech Communication*, 15(1–2):139–153.

Steedman, Mark. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

White, Michael, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.

White, Michael. 1995. Presenting punctuation. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 107–125.

White, Michael. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.