Decomposed scoring of CCG dependencies

Aditya Bhargava and Gerald Penn

Department of Computer Science University of Toronto Toronto, ON, Canada M5S 3G4 {aditya, gpenn}@cs.toronto.edu

Abstract

In statistical parsing with CCG, the standard evaluation method is based on predicate-argument structure and evaluates dependencies labelled in part by lexical categories. When a predicate has multiple argument slots that can be filled, the same lexical category is used for the label of multiple dependencies. In this paper, we show that this evaluation can result in disproportionate penalization of supertagging errors and obfuscate the truly erroneous dependencies. Enabled by the compositional nature of CCG lexical categories, we propose *decomposed scoring* based on subcategorial labels to address this.

To evaluate our scoring method, we engage fellow categorial grammar researchers in two English-language judgement tasks: (1) directly ranking the outputs of the standard and experimental scoring methods; and (2) determining which of two sentences has the better parse in cases where the two scoring methods disagree on their ranks. Overall, the judges prefer decomposed scoring in each task; but there is substantial disagreement among the judges in 24% of the given cases, pointing to potential issues with parser evaluations in general.

1 Introduction

With a suitably designed architecture, combinatory categorial grammar (CCG) supertaggers can learn to better maintain syntagmatic consistency, adjusting for their own errors to keep the sentence parsable (Vaswani et al., 2016; Bhargava and Penn, 2020). This kind of adjustment, however, comes at the expense of its evaluated word accuracy, which is the prevailing evaluation measure for supertagging. The standard, final *parser* evaluation is no kinder in such cases: it examines induced bilexical dependencies, but these are labelled (in part) by the lexical category assigned to the head word (Clark and

The code for decomposed CCG scoring is available online at https://www.cs.toronto.edu/~aditya/ccgds

Hockenmaier, 2002). If the category is incorrect, its outgoing dependencies are considered incorrect.

While other areas of NLP such as natural language generation and machine translation have recently warmed to efforts to validate their intrinsic evaluations against human judgements (e.g., Novikova et al., 2017; Reiter, 2018), this has not been the case so far with statistical parsing. This is likely because evaluating the quality of a syntactic parse requires grammatical expertise.

In this paper, we examine CCG parser evaluation and identify a number of cases where the standard CCG scoring method gives undesirable results. We address these shortcomings by introducing decomposed scoring. To evaluate our new method, we elicit judgements from categorial grammar (CG) experts in two pairwise selection tasks using Englishlanguage data from CCGbank (Hockenmaier and Steedman, 2007). In the first, intrinsic task, the judges are directly asked which of the two scoring methods they prefer for a given sentence. We find that they prefer decomposed scoring in 90% of the cases presented. In the second, extrinsic task, judges are given two different sentences, each with an erroneous parse, where the scoring methods disagree about which parse should be ranked higher. Here, we find that the judges do not have majority agreement in 24% of the cases presented; but where they do reach majority consensus, they agree with decomposed scoring in 62% of cases. The high disagreement raises important questions about statistical parser evaluations, which we discuss.

2 Background

Following Clark and Hockenmaier (2002), the standard evaluation measure for CCGbank-based CCG parsers is F_1 over bilexical dependencies. Each dependency represents a predicate-argument relationship as indicated by the corresponding lexical categories. A (labelled) CCG dependency *d* is defined as a 4-tuple $d = (h_p, h_a, c, s)$ where:



Figure 1: A CCGbank sentence, with ground truth shown above it and predictions by C&C underneath. Category features (e.g., s_{del}) are omitted for compactness. Errors are shown in orange, with erroneous dependency edges drawn dashed. Ground-truth dependencies marked as erroneous convey their absence from the prediction.

- $h_{\rm p}$ is the head word token of the predicate;
- $h_{\rm a}$ is the head word token of the argument;
- c is the lexical category of the predicate; and
- *s* is the predicate's argument slot number that is filled by the argument.

Given an input sentence, the set of corresponding ground-truth dependencies \mathcal{D}_G from CCGbank, and a candidate set of dependencies \mathcal{D}_C from a parser, the candidate dependencies are evaluated according to the F₁ score between the two sets:

$$F_{1}(\mathcal{D}_{G}, \mathcal{D}_{C}) = \frac{2|\mathcal{D}_{G} \cap \mathcal{D}_{C}|}{|\mathcal{D}_{G}| + |\mathcal{D}_{C}|}$$

A dependency $d \in \mathcal{D}_{G} \cup \mathcal{D}_{C}$ is considered correct if and only if $d \in \mathcal{D}_{G} \cap \mathcal{D}_{C}$. In computing $|\mathcal{D}_{C} \cap \mathcal{D}_{G}|$, individual dependency elements are compared for equality. Formally, for a given dependency $d = (h_{p}, h_{a}, c, s)$, let $d_{h} = (h_{p}, h_{a}), d_{c} = c$, and $d_{s} = s$. Then $\mathcal{D}_{C} \cap \mathcal{D}_{G}$ is:

$$\mathcal{D}_{\rm G} \cap \mathcal{D}_{\rm C} = \begin{cases} (g,c) & g \in \mathcal{D}_{\rm G} \land c \in \mathcal{D}_{\rm C} \\ \land g_{\rm h} = c_{\rm h} \\ \land g_{\rm c} = c_{\rm c} \\ \land g_{\rm s} = c_{\rm s} \end{cases}$$
(1)

This dependency-based measure directly evaluates the parser's ability to produce the intended predicate-argument structure. Models that analyze sentences with different derivations than the one provided by the corpus will not be penalized unless the derivation alters the semantics—i.e., this evaluation is invariant to spurious ambiguities.

3 Decomposed dependency scoring

In this paper, we do not take issue with the treatment of the head words h_p and h_a , but rather of the lexical category, *c*, and argument slot number, *s*. We alter how CCG dependency labels are compared so that a dependency's correctness no longer requires the entire lexical category to be correct and allow (judicious) flexibility in valid values for the slot number. As these modifications are dependent on subcategorial decompositions of the lexical category labels, we term the overall approach **decomposed scoring**.

3.1 Subcategorial labelling

Requiring predicted lexical categories to be *fully* equal to the ground truth can cause errors to be overpenalized. In the example shown in Figure 1, the parser makes a complement-adjunct confusione rror (a common parser pathology): the complement *in* is mistakenly analyzed as an adjunct. While this error is directly indicated by the erroneous dependencies between the verb and its complement, the standard scoring method "delocalizes" the error, marking 75% of the dependencies as erroneous.

To address this, we propose **subcategorial labelling** of CCG dependencies: instead of the entire lexical category, only the subcategory corresponding to the argument slot is used for comparison. To define this more formally, we first define a function $\arg_n(x)$ that extracts the subcategory for argument slot *n* from category *x*:

$$\arg_n(x) = \begin{cases} x & \text{if } n = \operatorname{arity}(x) = 0, \\ z & \text{if } x = (y|z) \land n = \operatorname{arity}(x), \\ \arg_n(y) & \text{if } x = (y|z) \land n < \operatorname{arity}(x), \end{cases}$$

where arity(*x*) is the number of arguments that *x* takes before yielding its target category and | is a variable that ranges over the categorial slash operators {/, \}. For example, $\arg_1(s/(s \setminus NP)) = s \setminus NP$.

Thus, subcategorial labelling replaces $g_c = c_c$ from Equation 1 with $\arg_{g_s}(g_c) = \arg_{c_s}(c_c)$. Returning to our example, subcategorial labelling allows the verb-subject dependency (from *believe* to *I*) to be marked correct as shown in Figure 2.

3.2 Subcategorial alignment

On its own, there are some situations where subcategorial labelling is insufficient. insufficient. In particular, subcategorial labelling is ineffective when the argument slot numbers differ between two dependencies that are being compared, as is the case for the prepositional complement dependency in Figure 2 (from *in* to *system*). While subcategorial labelling indicates that the argument subcategory is



Figure 2: The example from Figure 1 altered to use subcategorial labelling.

correct, the differing slot numbers mean that the dependency is still considered incorrect, even though the parser found the correct syntactic relationship.

We thus propose **subcategorial alignment**: we allow the proposed slot number to be considered correct if there exists a *plausible alignment* (to be defined shortly) between its corresponding argument slot in the candidate lexical category and the correct argument slot in the ground-truth lexical category.

In order to establish such an alignment, we first decompose the (full) lexical categories for the given dependencies into a linear representation consisting of its target category and its "directed" argument subcategories such that the distinction between left and right arguments is maintained. We call these linear representations **functorial sequences**. More formally, the functorial sequence fs(x) of category *x* is defined as:

$$fs(x) = \begin{cases} [x] & \text{if arity}(x) = 0\\ fs(y) \oplus [z] & \text{if } x = y|z, \end{cases}$$

where \oplus denotes list concatenation. For example, PP/NP and ((s\NP)\(s\NP))/NP decompose into the functorial sequences [PP, /NP] and [s, \NP, \(s\NP), /NP], respectively.

Next, we compute the Levenshtein distance between the two functorial sequences and then backtrack, gathering the set of optimal paths. A **plausible alignment** is then any match state (i.e., zerocost substitution) on any optimal path.

Formally, let $\mathcal{M}(x, r)$ denote the set of all match states (i, j) in any Levenshtein alignment between functorial sequences fs(x) and fs(r), where *i* indexes over fs(x) and *j* indexes over fs(r). Thus, in Equation 1, subcategorial alignment replaces $g_s = c_s$ with $(g_s, c_s) \in \mathcal{M}(g_c, c_c)$.

Our use of Levenshtein alignments is motivated by the view of many supertagging errors as insertions or deletions of categorial arguments; for example, in Figure 1, we see the /PP argument as having



Figure 3: The example from Figure 2 altered to use subcategorial alignment.

been deleted in the predicted category for *believe*. Levenshtein alignment is robust here as subcategorial alignment is only relevant when the other elements of the dependency tuple are correct.¹ As well, the alignment's monotonicity prevents swapped arguments (e.g., swapped direct and indirect objects) from mistakenly being marked correct.²

Returning to the complement-adjunct confusion example, the only plausible alignment that exists between PP/NP and ((s NP) (s NP))/NP is at the /NP directed subcategories. The dependencies from *in* to *system* are thus marked correct since there is a plusible alignment between the corresponding argument slots. As shown in Figure 3, this leaves only the dependencies that directly indicate the complement/adjunct relations as erroneous.

Other than complement-adjunct confusion, subcategorial alignment is also useful for prepositional phrase attachment errors, another common parser pathology. Refer to Appendix A.1 for an example.

3.3 Root node inclusion

Our final modification is the inclusion of root nodes. The choice of root is relevant in *de dicto–de re* distinctions (*inter alia*), as shown in Figure 4. Despite the error in the parse, the standard CCG evaluation assigns a perfect F_1 score since it does not include root nodes (and corresponding dependencies).

Importantly, including root nodes also addresses a potential pathology of directly using subcategorial labels. If the only error is in the choice of spanning category for the sentence, subcategorial labelling without a root can result in a perfect F_1 score since

^{1.} For instance, if a parser mistakenly uses a ditransitive verb category $((s\NP)/NP)$ rather than the correct transitive verb category $(s\NP)/NP$, a plausible alignment exists for both the direct and indirect object dependencies. But since the dependency endpoints differ, both cannot be marked correct.

^{2.} Such swapped arguments can occur if the parser incorrectly decides to use type-raising and backward crossing composition. Notably, unlabelled dependencies are unable to find such errors.



Figure 4: A sentence where the supertagger made no mistakes, but the parser's analysis indicates a *de re* reading instead of the correct *de dicto*.

the root category does not fill any argument slots. Including a root dependency that specifies the correct top-level category entirely addresses this. Refer to Appendix A.2 for an example.

4 Evaluating decomposed scoring

Thus far, we have argued for decomposed scoring on the basis of examples that show cases where decomposed scoring is able to more precisely isolate and penalize parsing errors. Our next aim is to determine the extent to which this capability holds true over a larger set of parser outputs when evaluated by expert judges in a systematic manner.

From here, we refer to the standard CCG evaluation as F_1 and to decomposed scoring as DF_1 .

4.1 Intrinsic and extrinsic evaluation tasks

The evaluation is split into two judgement tasks. The first task gauges whether CG researchers agree that DF₁ is better able to isolate parsing errors than F_1 . Judges are given sentences with corresponding dependency sets where F_1 and DF₁ disagree about the correctness of at least one dependency. Each sentence is presented as a pair of dependency figures similar to those presented above (e.g., Figures 1 and 3 are one such pair, though all figures in the judgement tasks include root nodes). For each sentence, the judges are asked which of the two methods better isolates the error made by the parser, as judged against the ground truth. As the first task directly compares F_1 and DF₁ on common sentences, we consider it to be an **intrinsic** evaluation.

The second task uses F_1 and DF_1 *in situ* as scoring methods; we therefore consider it to be an **extrinsic** evaluation. Since there is no objectively correct score for partially correct dependency sets, whether the score assigned by DF_1 to a given dependency set is better than that assigned by F_1 to the same set can only be evaluated in relative terms. At the extreme,

if DF_1 were to yield a different score than F_1 for each dependency set of interest but the two induced the same preorder over the dependency sets, the two methods would not be meaningfully different.

The second task therefore examines *pairwise* rank inversions: pairs of sentences where F_1 and DF_1 disagree on which sentence's dependency set is better. Here the judges are given pairs of different sentences and then asked to select the sentence that they believe has the better parser-generated dependency set.³ The question underlying the extrinsic task is thus whether using DF_1 instead of F_1 results in sentence rankings that better match those that would be assigned by human judges.

For both tasks, the instructions include wording directing the judges to consider semantics in their evaluations. This maintains the assumption inherent in evaluations based on predicate-argument structure that parsing errors are significant in proportion to their effects on semantics.

4.1.1 Task administration

The tasks are administered sequentially, with the second task being given to each judge after completion of the first. For each task, each judge is given a unique link to an online interface that describes notational conventions, specifies the task they are to complete, presents the data, and records their responses. No time limit is imposed, and previous judgements can be changed at any time.

Prior to having external judges complete the tasks, we conducted an internal pilot study; consult Appendix B for details. Here we focus on details and results of the main study only.

We use three parsers to generate parse predictions for sentences from the CCGbank test set: EasyCCG Lewis and Steedman (2014), C&C (Clark and Cur-

^{3.} As the figures are too wide to comfortably include here, refer to the supplementary materials of this paper for examples.

	Dev set		Test set	
Parser	F ₁	DF ₁	F ₁	DF ₁
C&C	83.4	88.5	84.2	88.9
EasyCCG	82.6	88.0	83.1	88.1
DepCCG	89.9	93.3	89.8	93.0

Table 1: F₁ and DF₁ scores of three parsers on CCGbank.

ran, 2007), and DepCCG (Yoshikawa et al., 2017). Table 1 shows the performance of the three parsers according to both F_1 and DF_1 . Verbatim copies of all data and instructions as given to each judge are available in the supplementary materials of this paper, including the judges' responses. Refer to Appendix C for further details of the data generation and sampling procedures.

4.1.2 Judge recruitment and compensation

To decrease the likelihood of bias towards DF_1 for the main study, we do not provide any judgements ourselves; nor do we ask members of our institution to do so. Instead, we sought unaffiliated CG researchers to provide their expert judgements: four judges were recruited via professional connections. Two judges were compensated at a flat rate and the remaining two were seconded by their employer. All four judges have peer-reviewed publications at relevant CG research venues and are fluent in English.

5 Results and discussion

In the first task, the judges ruled strongly in favour of DF₁, agreeing with it in ¹⁸/₂₀ cases. This is statistically significant (binomial test, $p \approx 2.0 \times 10^{-4}$). We therefore conclude that DF₁ is better than F₁ at identifying the ultimate error in the parser's output.

In the second task, we find some disagreement among judges: for ¹¹/₄₅ sentence pairs, two judges agreed with DF₁, while the remaining two disagreed with it, leading to a tie. But of the remaining 34 sentence pairs, a majority of judges agreed with DF₁ on 21 pairs. Even with the ties, this is also statistically significant ($p \approx 0.02$); we thus conclude that DF₁ is preferable to F₁.

Disagreement among judges in the second task merits discussion. Judges are taken to be providing ground truth, so tied pairs are cases where a ground truth judgement is unavailable. What does this mean for decomposed scoring, and for CCG parser evaluations more generally?

Concerning decomposed scoring, the first task's results empirically validate the utility of DF_1 over

 F_1 . D F_1 prevents obfuscation of erroneous dependencies, improving the granularity of the evaluation measure. In addition, there is the possibility of helping with the *training* of statistical parsers: when a dependency's only crime is sharing a lexical category with an erroneous one, training a parser to learn that both are errors may cause it to learn to avoid correct dependencies.

Moreover, we examined the sentence pairs in the second task ourselves and found that even when we judged DF₁'s rank inversion to be incorrect, undoing DF₁'s changes would not address the issue; the severity of parsing errors varies and is in part modulated by semantic intricacies. We thus expect that the disagreements among judges are due at least in part to underlying differences in opinions about sentence meaning and/or salience.

Turning to the broader issue of CCG parser evaluations, the lack of definitive ground truth for many inter-sentence comparisons implies limitations for inter-parser comparisons. Imagine a case where parser A claims to outperform parser B, and closer inspection reveals that A and B differ in their outputs on only two sentences. For one of these sentences, A's output has a higher F_1 than does B's; for the other, the opposite is true, but the difference in F_1 is smaller. Now, the claim that A outperforms B becomes a claim that the parse that A produces for the first sentence is better than the parse that B produces for the second. And yet, as indicated by the judges' disagreements in the second task, it is not always possible to make these kinds of judgements.

6 Conclusion

We have found that the standard CCG evaluation method's choice of dependency labels is prone to amplifying minor errors. We proposed decomposed scoring and validated it by consulting experts. From their judgements, we conclude that decomposed scoring is better at isolating parser errors and is, overall, a better choice than the standard scoring method. Disagreement in the second task, however, is a source of concern and suggests potential issues for CCG parser evaluations. Given the frequentlysmall deltas between modern parsers, this is worth investigating further.

7 Limitations

While we used multiple parsers to avoid biasing the evaluation towards one parser, all parsers used are relatively high-performing parsers—all have labelled F_1 scores above 0.8 on the CCGbank development and test sets. This evaluation is thus biased towards especially difficult sentences, since those will be the ones where good parsers produce errors. While we found no correlation between parser scores and judge disagreement, at least suggesting that the judgements were not a function of parse quality, poorer parsers (or good parsers on novel domains) may make different kinds of errors than those that appeared in our sample. It is unclear how F_1 and DF₁ would compare under such circumstances; understanding this better remains an open area of research.

The relatively high disagreement among judges in the second task (24% of sentence pairs) is concerning, but it should be noted that the sentence pairs were sampled from a set of disagreements between two different *scoring methods*. The extent to which this is a problem in practice is unclear, as judge agreement may not be as low on outputs from different *parsers* evaluated by the *same* scoring method—but it could also be lower.

Although the dependency-based evaluations discussed in this paper are standard for CCGbankbased statistical CCG parser evaluations, the reliance on extra resources (namely, the generate program and markup files from C&C) makes it somewhat unique. Because of this, the extent to which decomposed scoring, or the ideas behind it, would be useful for other evaluations scenarios (such as for other corpora, including CCG corpora from other languages) is unclear.

Acknowledgements

We thank Umut Özge, Jakob Prange, Laura Rimell, and Miloš Stanojević for participating in our judgement task, Jinman Zhao and Timothy Fowler for their judgements and feedback during our pilot study, and our anonymous reviewers for their feedback. We gratefully acknowledge Google Deep-Mind for seconding research staff in support of this project.

References

- Aditya Bhargava and Gerald Penn. 2020. Supertagging with CCG primitives. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 194–204, Online. Association for Computational Linguistics.
- Stephen Clark and James R. Curran. 2007. Widecoverage efficient statistical parsing with CCG and

log-linear models. *Computational Linguistics*, 33(4):493–552.

- Stephen Clark, Darren Foong, Luana Bulat, and Wenduan Xu. 2015. The Java version of the C&C parser version 0.95. Technical report, University of Cambridge Computer Laboratory.
- Stephen Clark and Julia Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *LREC Beyond PAR-SEVAL Workshop*, pages 60–66, Las Palmas, Spain.
- 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.
- Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.
- Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.
- Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.
- Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with LSTMS. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 232–237, San Diego, California. Association for Computational Linguistics.
- Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada. Association for Computational Linguistics.

A Additional examples

A.1 Prepositional phrase attachment

As is the case for complement-adjunct confusion errors, prepositional phrase attachment errors are affected by argument slot renumbering, particularly when the analysis is as an adjunct, but there is ambiguity in whether the adjunct is modifying a noun or a verb. Since nouns and verbs have different arities, any arguments for their corresponding adjuncts will end up with different slot numbers, as shown in Figure 5.



Figure 5: A CCGbank sentence parsed by C&C, which makes a prepositional phrase attachment error.



Figure 6: The example from Figure 5 altered to use decomposed scoring.



Figure 7: A CCGbank sentence parsed by DepCCG.

Decomposed scoring again localizes the error directly to the adjuncts and their (adjunctival) arguments, as shown in Figure 6.

A.2 Root category pathology

In Figure 7, the standard CCG evaluation substantially over-penalizes the very minor error in the root category ($s_{ng} vs s_b$).

However, without including root nodes, subcategorial labelling suffers from a pathology in such cases. As shown in Figure 8, the subcategorial labelling (alone) results in all dependencies being



Figure 8: The example from Figure 7 altered to use subcategorial labelling.

marked correct and thus a perfect F_1 score, despite the error in the parse.

Fortunately, as shown in Figure 9, adding the root node entirely solves this issue.

B Pilot study

Before our main study, we first conducted a pilot study to confirm the details of our tasks. Two members of our research lab served as pilot judges: one professor (the second author of this paper) and one Ph.D. student (uninvolved with the work in this pa-



Figure 9: The example from Figure 8 altered to include root nodes, which is effectively full decomposed scoring in this case.

per), both of whom have peer-reviewed publications in relevant CG research venues.

For both tasks in the pilot study, all judges were given the same data for evaluation. The first task included 10 annotation items (sentences) while the second task included 20 annotation items (sentence pairs). See Appendix C for details of the data generation and sampling procedures.

B.1 Results

In the first task, opinion among the judges was unanimous: for each sentence, both judges agreed that the labelling and error assignment provided by DF₁ was better at identifying the ultimate error in the parser's output. For the null hypothesis of judges that make their selections at (uniform) random, the binomial test indicates that this degree of agreement in favour of DF₁ is extremely unlikely, with $p \approx 9.5 \times 10^{-7}$; we therefore reject it.

In the second task, the judges agreed with each other only half the time (1% pairs). Out of the ten cases where they agreed with each other, however, they agreed with the DF₁ ranking nine times. We can again reject the null hypothesis of judges choosing at (uniform) random: the binomial test indicates that such high agreement in favour of DF₁ would have $p \approx 0.04$.

B.2 Changes for main study

The results of the pilot study led us to make the following alterations for the main study:

• Given the complete agreement between judges in the first task, we treated judges as interchangeable for the first task in the main study. Each judge was thus given a *different* sample of sentences, allowing more sentences to be covered. • To account for the high level of inter-judge disagreement in the second task, we increased the number of sentence pairs in the task to 45; the number of sentences in the first task was reduced to five per judge in order to make better use of the judges' time. As with the pilot study, each judge was given the same set of sentences.

As well, since each parser was tuned on the CCGbank development set, we used the CCGbank test set (i.e., section 23) to sample the sentences for the main study. Remaining task and administration details were the same as for the pilot study.

C Data generation and sampling

To generate the data for both tasks, we started with the CCGbank development set (i.e., section oo). As the dependency figures can easily become very wide, we excluded all sentences where the ground truth has more than 20 dependencies (including the root dependency). This ensured that the figures fit legibly on most displays. Next, we ran off-theshelf parsers on the remaining sentences to produce predicted parses. For the pilot study, we used EasyCCG Lewis and Steedman (2014) only. For the main study, we prevented the results from being biased towards a single parser by using three different parsers: EasyCCG, C&C (Clark and Curran, 2007), and DepCCG (Yoshikawa et al., 2017). When sampling data for the two tasks, the parsers were alternated for selection per sampled item so that each parser was evenly represented in the data given to the judges (15 sentence pairs each for the second task). We converted all parser outputs into dependency sets using the generate program from C&C; we used updated markup files from the Java version of C&C (Clark et al., 2015). Sentential categories were extracted as needed from the parsers' outputs, after which each sentence was scored with both F₁ and DF₁. From here, the process diverged for the first and second tasks.

For the first (intrinsic) task, we kept only those sentences that met the relevant criterion: F_1 and DF_1 must disagree about at least one dependency. From these, we sampled sentences uniformly and without replacement to yield the set of sentences to be presented to judges for the first task (10 sentences for the pilot study and 5 for the main study). Although F_1 does not evaluate the root dependency, we found that omitting the root node from one of the two dependency figures in each pair made for a visually conspicuous absence; instead, we kept the root node for both cases, labelled the dependency accordingly, and never marked the root dependency as erroneous for the F_1 diagrams.

For the second (extrinsic) task, we first gathered all sentence pairs where F_1 and DF_1 disagreed on the relative ranks of the two pair elements. In order to avoid differences in scale, we then removed all pairs where the two sentences in the pair did not have the same number of dependencies in their ground truths. Next, in order to keep the task as simple as possible, we disallowed ties and therefore removed all pairs where at least one of F_1 or DF_1 assigned the same score to both elements in a pair. From these, sentence pairs were sampled uniformly and without replacement to yield the set of sentence pairs to be presented to judges for the second task (20 for the pilot study and 45 for the main study).

ACL 2023 Responsible NLP Checklist

A For every submission:

- A1. Did you describe the limitations of your work? *Section 7*
- □ A2. Did you discuss any potential risks of your work? *Not applicable. Left blank.*
- A3. Do the abstract and introduction summarize the paper's main claims? *Section 1*
- A4. Have you used AI writing assistants when working on this paper? *Left blank.*

B ☑ Did you use or create scientific artifacts?

Section 4

- B1. Did you cite the creators of artifacts you used? Section 1
- □ B2. Did you discuss the license or terms for use and / or distribution of any artifacts? *Not applicable. Left blank.*
- □ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)? *Not applicable. Left blank.*
- □ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it? *Not applicable. Left blank.*
- B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
 Not applicable. Left blank.
- B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be. *Section 4 and Appendix C*

C Z Did you run computational experiments?

Left blank.

□ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used? *No response.*

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

- □ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
 No response.
- □ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run? *No response.*
- C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

No response.

- **D** Did you use human annotators (e.g., crowdworkers) or research with human participants? *Section 4 and Appendices B and C*
 - D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
 Supplementary materials
 - D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?

Most details provided in Section 4.1.2. Payment details omitted as it qualifies as an exception under our university's ethics protocol as collegial review.

- D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used? Did not discuss in paper but participants were clear on this when recruited.
- ☑ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board? *Exempt*
- ✓ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data? Section 4.1.2