1 Introduction

Combinatory Categorial Grammar (CCG) (Steedman, 2000) is a mildly context-sensitive grammar formalism. Several neural CCG parsing methods have been proposed so far (Lewis and Steedman, 2014; Xu et al., 2015; Lewis et al., 2016; Vaswani et al., 2016; Lee et al., 2016; Xu, 2016; Yoshikawa et al., 2017; Stanoević and Steedman, 2019, 2020; Bhargava and Penn, 2020; Tian et al., 2020; Prange et al., 2021; Liu et al., 2021). Currently, neural span-based models (Cross and Huang, 2016; Stern et al., 2017; Gaddy et al., 2018; Kitaev and Klein, 2018) have been successful in the field of constituency parsing. However, we cannot directly apply this technique to CCG parsing. Span-based models assume that each node label in parse trees can be predicted independently, while, in CCG, each node label (category) is strictly restricted by CCG rule schemata. The independence assumption of span-based models implies that the models are not guaranteed to generate valid CCG derivations. To solve this problem, we propose a method of representing CCG derivations in a way suitable for span-based parsing models. Our proposed representation decomposes CCG derivations into several independent pieces and prevents the span-based parsing models from violating the CCG rule schemata. This characteristic is not suitable for span-based parsing models because they predict node labels independently. In other words, span-based models may generate invalid CCG derivations that violate the rule schemata. Our proposed representation decomposes CCG derivations into several independent pieces and prevents the span-based parsing models from violating the schemata. Our experimental result shows that an off-the-shelf span-based parser with our representation is comparable with previous CCG parsers.

2 CCG and Span-based Parsing

This section gives an overview of Combinatory Categorial Grammar (CCG) (Steedman, 2000) and explains why we cannot directly apply the span-based approach to CCG parsing.

2.1 Combinatory Categorial Grammar

CCG represents syntactic information by basic categories (e.g., S, NP) and complex categories. Complex categories are in the form of \(X/Y\) or \(X/Y\), where \(X\) and \(Y\) are categories. Intuitively, each category \(X/Y\) means that it receives a category \(Y\) from its right and returns a category \(X\). In the case of \(X/Y\), the direction is from its left. Formally, categories are combined using CCG rule schemata. Figure 1 shows CCG rule schemata. Here, \(X, Y, Z_1, \ldots, Z_d\) are categories, and \(i \in \{/, \}\). \(|Z_1\ldots|Z_d\) is called an argument stack (Kuhlmann and Satta, 2014), and we use a Greek letter to represent an argument stack.
rule schema:
\[ X/Y \Rightarrow X\alpha. \]  
(1)

We define \(|\alpha| = d\) and the arity of a category \(Y = X\alpha\) where \(X\) is a basic category is defined as follows:
\[ \text{arity}(Y) = |\alpha| \]  
(2)

2.2 Span-based Parsing

A span-based parsing model (Stern et al., 2017; Gaddy et al., 2018; Kitaev and Klein, 2018) has a single scoring function \(s(i,j,l)\) that scores each label \(l\) for each span \((i,j)\). The score of a tree \(T\) is defined as follows:
\[ s(T) = \sum_{(i,j,l) \in T} s(i,j,l). \]  
(3)

The parsing problem is formulated as finding the tree \(T^*\) with the highest score:
\[ T^* = \arg\max_T s(T) \]  
(4)
and can be solved using an efficient CKY-like parsing algorithm because of the following characteristic:

- The model can determine each label \(l\) for a span \((i,j)\) independently of the other spans.

Unfortunately, CCG parsing cannot take this approach because each label (category) is strictly restricted by the CCG rule schemata. If we apply the span-based approach to CCG parsing forcibly, the following problem occurs:

- The parsing model may generate invalid CCG derivations that violate the CCG rule schemata.

3 Span-based representation

To overcome the problem described in the previous section, we propose a new representation for CCG derivations. We call the new ones span-based representations (SBRs for short), which decomposes CCG derivations into several independent pieces to prevent the span-based parsing model from violating the CCG rule schemata. Figure 2 shows an example of CCG derivation and its SBR version.

We realize span-based CCG parsing as follows:

1. Convert CCG derivations into SBRs (Section 3.2).
2. Train a span-based parsing model using SBRs and parse sentences to generate SBRs.
3. Convert the output SBRs into CCG derivations. (Section 3.3).

The basic idea behind our method is that each node label in an SBR represents a constraint on the categories of nodes in a CCG derivation. Our method recovers a CCG derivation from its SBR version by satisfying such constraints. Because constraints encoded in SBR’s labels are independent, a span-based model using SBRs does not suffer from violating CCG rule schemata.

3.1 SBR’s label

An SBR’s label consists of the following information:

- a CCG rule schema
- a mapping from variables that occur only in the left-hand side of the rule to categories

For each node \(n\) (except leaf nodes) in a CCG derivation, its SBR version has a corresponding node. The SBR’s label means that the category of \(n\) is created by the specified rule schema, and the categories of \(n’s\) children satisfy the constraint represented by the mapping. For example, the label (\(>^0, Y := \text{NP}\)) means that the left and right children’s categories are in the form of \(X/\text{NP}\) and \(\text{NP}\) and \(X\) is inherited from its parent’s category.

3.1.1 Additional information

SBR’s label cannot encode root categories of CCG derivations and unary rules. To encode this information, we introduce three types of additional information:

- \(\text{RT} : X\) means that the category of the node \(n\) is \(X\), if \(n\) is the root node.
- \(\text{UL} : X\) means that the left child \(l\) is unary branching and the category of \(l’s\) child is \(X\).
- \(\text{UR} : X\) means that the right child \(r\) is unary branching and the category of \(r’s\) child is \(X\).

We call these information tags.
Algorithm 1 recovers a CCG derivation from an SBR. This process is repeated recursively until the leaf nodes are reached. When the SBR’s label is in the form of \( (>^d, \ldots) \) or \( (<^d, \ldots) \) and \( \text{arity}(P) < d \), \( L \) and \( R \) cannot be defined. In this case, \( \text{recoverCAT}(S, P) \) replaces \( d \) with the minimum possible value for which \( \text{arity}(P) < d \).

### 4 Generating OOV categories

In our proposed representation, lexical categories are not directly assigned to words. Lexical categories are decomposed into several node labels. This means that lexical categories are not defined by a finite set and that the span-based parsing model learned from SBRs may generate OOV lexical categories that do not appear in the training data.
5 Experiment

We conducted an experiment using the CCGBank (Hockenmaier and Steedman, 2007) to evaluate the performance of our method. We used the Berkeley Neural Parser (Kitaev and Klein, 2018) with BERT (Devlin et al., 2019) as a span-based parser. We converted the training (sections 02–21) and the development (section 00) data into SBRs and learned the model from the data. The number of SBR’s labels in the training data was 486. The hyperparameters for training were identical to those of Kitaev et al. (2019). We evaluated the parsing performance using the C&C parser’s generate program (Clark and Curran, 2007). As a baseline model, we trained a model using the CC derivations.

Table 2 shows parsing performances on the test data. Our proposed and the baseline methods have high precision (92.8% and 90.9%) but low recall (82.2% and 90.7%). One of the reasons for the low recall was that the C&C parser’s generate program failed to obtain dependencies from the output CC derivations. Our proposed and the baseline methods failed to obtain dependencies from 206 and 371 sentences of 2407 test data sentences, respectively. The generate program cannot work when the CCG derivation is invalid or has a lexical category that is not listed in its markedup file.

Table 3: Recall for OOV lexical categories on the test data.

To mitigate this problem, we added such lexical categories to the markedup file. Adding lexical categories increased the recall (87.6%) of our method significantly. On the other hand, the recall of the baseline method was still low (76.8%) due to the invalid CCG derivations. This result shows that a span-based parsing using CC derivations does not work well and that our proposed method improves the parsing performance. The final result of our method was comparable with previous CCG parsers.

5.1 OOV categories

Another interesting point of our method is the possibility of generating OOV categories. Table 3 shows the recall for OOV lexical categories. We obtained a similar result with previous research. Our method correctly assigned OOV categories for 4 words.

The new markedup file was generated automatically.

There are only 22 occurrences of OOV categories in the test data.

5.2 Recovering OOV categories

Table 2: Labeled F1 on the test data.
We can say that our proposed approach can treat OOV categories.

6 Conclusion

This paper proposed a new representation for CCG derivations. Our proposed representation realizes a span-based CCG parser that follows the CCG binary rule schemata. Furthermore, the parser can generate OOV categories. One remaining problem in the proposed method is to treat unary rule schemata in CCG. Our method encodes unary rules using the additional information described in Section 3.1.1, but this approach may violate the unary rule schemata. In the future, we will extend the method to treat CCG unary rules validly.

Acknowledgements

We thank anonymous reviewers for their helpful comments.

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. Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics, 33(4):493–552.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What’s going on in neural constituency parsers? an analysis. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

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

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Marco Kuhlmann and Giorgio Satta. 2014. A new parsing algorithm for Combinatory Categorial Grammar. Transactions of the Association for Computational Linguistics, 2:405–418.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2366–2376, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 221–231, San Diego, California. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with semi-supervised supertagging. Transactions of the Association for Computational Linguistics, 2:327–338.

Yufang Liu, Tao Ji, Yuanbin Wu, and Man Lan. 2021. Generating CCG categories. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 13443–13451.

Jakob Prange, Nathan Schneider, and Vivek Srikrum. 2021. Supertagging the Long Tail with Tree-Structured Decoding of Complex Categories. Transactions of the Association for Computational Linguistics, 9:243–260.

Miloš Stanojević and Mark Steedman. 2019. CCG parsing algorithm with incremental tree rotation. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 228–239, Minneapolis, Minnesota. Association for Computational Linguistics.
Miloš Stanojević and Mark Steedman. 2020. Max-margin incremental CCG parsing. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4111–4122, Online. Association for Computational Linguistics.

Mark Steedman. 2000. The Syntactic Process. The MIT press.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Categorial Grammar with attentive graph convolutional networks. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6037–6044, Online. Association for Computational Linguistics.

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.

Wenduan Xu. 2016. LSTM shift-reduce CCG parsing. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1754–1764, Austin, Texas. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 250–255, Beijing, China. 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.