Holographic CCG Parsing

Ryosuke Yamaki **Tadahiro Taniguchi**
Ritsumeikan University
1-1-1 Noji Higashi, Kusatsu, Shiga Japan
{yamaki.ryosuke,taniguchi}
@em.ci.ritsumei.ac.jp

Daichi Mochihashi
The Institute of Statistical Mathematics
10-3 Midori-cho, Tachikawa city,
Tokyo Japan
daichi@ism.ac.jp

Abstract

We propose a method for formulating CCG as a recursive composition in a continuous vector space. Recent CCG supertagging and parsing models generally demonstrate high performance, yet rely on black-box neural architectures to implicitly model phrase structure dependencies. Instead, we leverage the method of holographic embeddings (Nickel et al., 2016) as a compositional operator to explicitly model the dependencies between words and phrase structures in the embedding space. Experimental results revealed that holographic composition effectively improves the supertagging accuracy to achieve state-of-the-art parsing performance when using a C&C parser. The proposed span-based parsing algorithm using holographic composition achieves performance comparable to state-of-the-art neural parsing with Transformers. Furthermore, our model can semantically and syntactically infill text at the phrase level due to the decomposability of holographic composition.

1 Introduction

Combinatory Categorial Grammar (CCG; Steedman 2000) is a highly lexicalized grammar formalism comprising syntactically rich lexical categories and a limited number of combinatory rules. In principle, CCG is suitable for modelling complicated syntactic structures and operates as a natural interface connecting syntax to semantics because of its isomorphism with lambda calculus (Bos et al., 2004; Mineshima et al., 2015; Martinez-Gómez et al., 2016). In this paper, we propose a method to formulate CCG (a discrete symbol system) as an operation between distributed representations in a continuous vector space, demonstrating its contribution to improved supertagging performance and span-based parsing.

Prior studies on PCFG, compositional vector grammar (CVG; Socher et al. 2013a), and its generalization, latent vector grammar (LVeG; Zhao et al. 2018), have shown the efficacy of representing discrete symbols as vector operations. High-dimensional vectors' expressive power complements syntactic disambiguation, which is difficult to address solely through discrete symbols. We propose a model that bridges discrete symbols and continuous vectors in CCG.

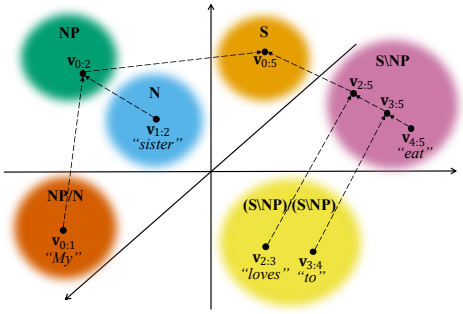

Figure 1: Conceptual diagram of holographic composition of vectors in embedding space according to CCG. Each pair of arrows represent a recursive composition of vectors without any additional parameters.

In this study, we introduce recursive vector composition in the embedding space illustrated in Figure 1 by employing holographic embeddings (HolE; Nickel et al. 2016) to incorporate syntactic structures into the supertagging and parsing model explicitly.

Similar methods for embedding tree structures into fixed-length vectors, CVG, and kernel-inspired encoders with recursive mechanisms for interpretable trees (KERMIT; Zanzotto et al. 2020a) have been proposed. Our model differs from CVG, as it does not require a large number of matrix parameters for nonlinear compositions, and directly optimizes parsing, enabling the construction of phrase-level representations by dynamically exploring phrase structures, whereas KERMIT requires an external parser.

Experiments revealed that phrase-level dependency modelling with holographic composition can induce correct supertagging, achieving state-of-the-

art performance in supertagging and parsing with a C&C parser (Clark and Curran, 2007; Clark et al., 2015), and further improved performance with a novel span-based parsing algorithm.[1]

Additionally, we focused on the fact that the inverse operation of holographic composition is easily available. This property can be applied to text-infilling tasks, predicting missing parts of sentences consistent with the rest syntactically and semantically. This task is difficult to accomplish using the existing neural architectures.

The main contributions of this research can be summarized as follows:

1. We introduce HolE as a recursive compositional operator for explicit modelling of syntactic structures, enabling CCG to be treated as an operation between distributed representations. This modelling improves supertagging and parsing, achieving state-of-the-art performance with a C&C parser.

2. We propose a novel span-based parsing algorithm incorporating phrase-level representation from our model, achieving comparable performance to the current state-of-the-art.

3. We propose an approach to compute phrase-level representations containing rich syntactic information while satisfying decomposability. We further demonstrate the applicability of decomposability to phrase-level text-infilling.

2 Background and Related Work

2.1 Recursive Compositional Models

Previous studies have shown benefits of explicit syntactic information incorporation into neural networks (Socher et al., 2011, 2013a,c; Tai et al., 2015; Zhu et al., 2015; Zhang et al., 2016; Zhao et al., 2018; Wang et al., 2019; Zanzotto et al., 2020a).

First, CVG (Socher et al., 2013a) was used to modify the recursive neural network (Socher et al., 2011), resulting in improved PCFG parsing performance. It recursively composes vectors of words and phrases using a nonlinear composition operation for a PCFG rule $C \rightarrow A\ B$ as

$$\mathbf{c} = \tanh\left(W_{C\rightarrow AB}\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}\right), \qquad (1)$$

where $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are d-dimensional vectors that represent A, B and C, respectively. $W_{C\rightarrow AB}$ is a $d \times 2d$ matrix for each rule, $C \rightarrow A\ B$. Therefore, it contains a huge number of parameters as well as word vectors themselves: when d is as small as 100 and there are 882 binary rules, as in (Socher et al., 2013a), it needs $100 \times 200 \times 882 = 17,640,000$ parameters for the matrix $W_{C\rightarrow AB}$, not to mention about the difficult nonlinear optimization involved.[2] For the same reason, Compositional Distributional Semantics (Polajnar et al., 2015), a method of composing phrase-level semantic representations, using tensors whose order is defined by CCG type, is hard to scale for higher dimensions.

A related study, KERMIT (Zanzotto et al., 2020a) is a model that embeds parse tree structures and subtrees in PCFG into fixed-length vector representations via recursive vector composition, enhancing the performance of downstream tasks. A comparison of this model with ours is given in Section 3.1.

2.2 Span-Based Parsing

Clark (2021) applied Transformers to span-score-based PCFGs (Stern et al., 2017; Kitaev and Klein, 2018) for CCG parsing, achieving significant performance gains. These studies computed a vector for each span in a sentence, input it into a feed-forward neural network to obtain a span score, and then apply it to a chart-based parsing algorithm.

Specifically, Kitaev and Klein (2018) calculated the vector $\mathbf{y}_{i:j}$ corresponding to the span from the ith word to the jth word as follows:

$$\mathbf{y}_{i:j} = [\overrightarrow{\mathbf{y}}_j - \overrightarrow{\mathbf{y}}_i; \overleftarrow{\mathbf{y}}_{j+1} - \overleftarrow{\mathbf{y}}_{i+1}], \qquad (2)$$

where $\overrightarrow{\mathbf{y}}_k$ and $\overleftarrow{\mathbf{y}}_k$ denote the right and left halves, respectively, when the vector $\mathbf{y}_k$ associated with the kth word is split by half. Constructing a vector of the span via simple subtraction between vectors, as in Equation (2), does not explicitly reflect the internal structures of the span.

3 Holographic CCG

Our research objective is to compute phrase-level representations to capture dependencies and hierarchical relationships among its internal components for supertagging and parsing in CCG. We describe the mechanism for composing these representations and discuss their application to supertagging. We then introduce a novel span-based parsing that utilizes these representations.

[1] Our implementation used for this paper is available at https://github.com/Ryosuke-Yamaki/Hol-CCG.git.

[2] Therefore, Socher et al. (2013a) reported that d should be as small as only 25 for stable training. Furthermore, CVG applies a nonlinear activation of a hyperbolic tangent, complicating optimization during training.

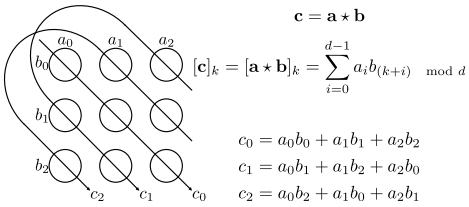

Figure 2: Schematic of circular correlation: adapted from Plate (1995); Nickel et al. (2016) (each circle representing a vector element, arrows denoting pattern of addition).

3.1 Holographic Embeddings

We explore methods to compose phrase-level representations capturing dependencies and hierarchical relationships between components. We focus on the commonalities between knowledge graphs and syntactic structures of natural language sentences. Both of them represent nonlinear relationships depending on the semantic aspect of each component, suggesting existing knowledge graph embedding methods (Socher et al., 2013b; Nickel et al., 2016; Trouillon et al., 2016; Abboud et al., 2020) are applicable to our objective. We employ HolE (Nickel et al., 2016) due to its desirable properties for embedding phrase structures without additional parameters, as we describe below.

HolE uses circular correlation (Plate, 1995) as a compositional operator for sophisticated knowledge graph modelling while maintaining computational efficiency. Focusing on HolE and circular correlation as compositional operators to model dependencies and hierarchical relationships, we compose two vectors $\mathbf{a}, \mathbf{b}$ into a single vector $\mathbf{c}$.

$$\mathbf{c} = \mathbf{a} \star \mathbf{b}, \tag{3}$$

where $\star : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ denotes a circular correlation:

$$[\mathbf{c}]_k = [\mathbf{a} \star \mathbf{b}]_k = \sum_{i=0}^{d-1} a_i b_{(k+i) \mod d}. \tag{4}$$

Circular correlation can be computed via Fourier transform, such as

$$\mathbf{c} = \mathbf{a} \star \mathbf{b} = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{a})} \odot \mathcal{F}(\mathbf{b})), \tag{5}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the fast Fourier transformation and its inverse, respectively, $\overline{\mathcal{F}(\cdot)}$ represents conjugation in a complex space, and $\odot$ denotes an element-wise product. Figure 2 shows a schematic of the circular correlation when $d = 3$.

Circular correlation exhibits desirable characteristics for our objective.

Noncommutative: Generally, $\mathbf{a} \star \mathbf{b} \neq \mathbf{b} \star \mathbf{a}$ holds true, making noncommutativity attractive for modelling asymmetric relations; for example, two noun phrases *"human right"* and *"right human"* will be composed into different vectors.

Nonassociative: Circular correlation is nonassociative, i.e., $(\mathbf{a} \star \mathbf{b}) \star \mathbf{c} \neq \mathbf{a} \star (\mathbf{b} \star \mathbf{c})$, making it ideal for modelling hierarchical structures. For example, the phrase *"saw a girl with a telescope"* yields different vectors when a circular correlation is used, depending on the internal structure *"((saw (a girl)) (with (a telescope)))"* and *"(saw ((a girl) (with (a telescope))))"*. Associative operations, however, yield the same representation, thus failing to reflect the internal structure.

Circular convolution, a similar operation to circular correlation, does not satisfy the above two properties:

$$[\mathbf{c}]_k = [\mathbf{a} * \mathbf{b}]_k = \sum_{i=0}^{d-1} a_i b_{(k-i) \mod d} \tag{6}$$

where $* : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ denotes the circular convolution operator. Circular convolution can be computed via Fourier transform, such as

$$\mathbf{c} = \mathbf{a} * \mathbf{b} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{a}) \odot \mathcal{F}(\mathbf{b})), \tag{7}$$

Here, KERMIT (Zanzotto et al., 2020a) utilizes shuffled circular convolution as the vector composition operator to guarantee the above properties.

$$\mathbf{c} = \mathbf{a} \otimes \mathbf{b} = \mathbf{a} * \Phi\mathbf{b}\,, \tag{8}$$

where $\otimes : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the shuffled circular convolution operator and Φ denotes a permutation matrix that shuffles the elements of $\mathbf{b}$. Section 5.3 provides an experimental comparison of these three operators. Another major difference between our model and KERMIT is that our model does not rely on an external parser to extract tree structure from the input.

Circular correlation is a first-degree noncommutative operation, making it difficult to distinguish between $\mathbf{c} \star (\mathbf{a} \star \mathbf{b})$ and $\mathbf{b} \star (\mathbf{a} \star \mathbf{c})$ (Zanzotto and Dell'Arciprete 2012). However, this is not critical for parsing, as it is sufficient to distinguish between possible internal structures of a given fixed word order sentence. We refer to the vector composition operation by circular correlation as holographic composition in this paper.

Zanzotto et al. (2020b) approximates the CKY algorithm using matrix multiplication and the property of holographic representation from Plate (1995). While similar to our approach in exploiting holographic representations and operations, our

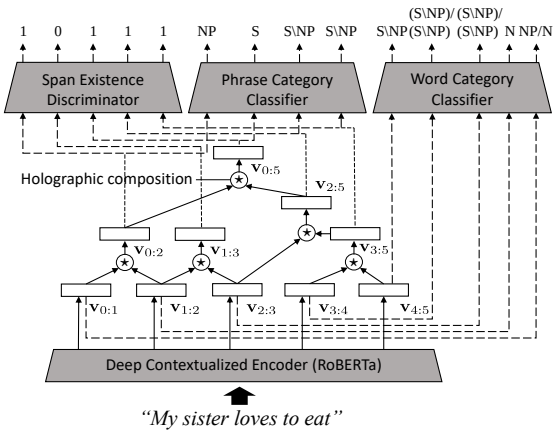

Figure 3: Proposed approach of composing phrase and sentence-level representation to predict categories and probability of span existence in Holographic CCG.

approach differs in performing a recursive holographic composition of distributed representations (described in the next section).

3.2 Recursive Vector Composition

We acquired phrase- and sentence-level representations via recursive composition of word representations, as illustrated in Figure 3 for the input sentence *"My sister loves to eat"*. First, the input sentence is fed into a RoBERTa encoder (Liu et al., 2019; Wolf et al., 2020), obtaining high-dimensional vectors $(\mathbf{v}_{0:1}, \ldots, \mathbf{v}_{4:5})$.

For a given phrase structure representable by an arbitrary binary tree, vector representations of the phrase and sentence are computed by applying holographic composition recursively. The vector for the entire sentence $\mathbf{v_{0:5}}$ is computed based on each word and phrase vector as follows:

$$\begin{aligned}\mathbf{v}_{0:5} &= \mathbf{v}_{0:2} \star \mathbf{v}_{2:5} \\ &= (\mathbf{v}_{0:1} \star \mathbf{v}_{1:2}) \star (\mathbf{v}_{2:3} \star \mathbf{v}_{3:5}) \\ &= (\mathbf{v}_{0:1} \star \mathbf{v}_{1:2}) \star (\mathbf{v}_{2:3} \star (\mathbf{v}_{3:4} \star \mathbf{v}_{4:5})).\end{aligned}$$

We observed a rapid norm increase of vectors with recursive holographic composition, thus necessitating norm constraint. We adopted either of two methods of norm constraint.

Normalization on real space: We imposed a norm constraint on all words and composed phrase vectors in real space.

$$\mathbf{v}' = k \cdot \frac{\mathbf{v}}{\max(\|\mathbf{v}\|, \epsilon)}, \tag{9}$$

where $\mathbf{v}$ and $\mathbf{v}'$ denote vectors without and with the imposed norm constraint, respectively, and k is the desired norm after normalization.

Complex unit magnitude projection: Ganesan et al. (2021) introduce a method applying a norm constraint to a vector in complex space as follows:

$$\mathbf{v}' = \mathcal{F}^{-1}\left(\cdots, \frac{\mathcal{F}(\mathbf{v})_i}{|\mathcal{F}(\mathbf{v})_i|}, \cdots\right), \tag{10}$$

where $\mathcal{F}(\mathbf{v})_i$ denotes the ith element of a vector mapped into a complex space by a Fourier transformation. Applying the norm constraint to word vectors yields a norm of 1 for all composed vectors, avoiding rapid norm increase. Furthermore, this yields desirable properties, such as decomposability (described in Section 3.5).

3.3 Supertagging

In CCG, supertagging is the task of assigning a plausible CCG category to each word in the sentence. In existing supertagging methods, various encoders transform each word into a high-dimensional vector that is input to the classifier to predict the appropriate category for each word (Vaswani et al., 2016; Lewis et al., 2016; Tian et al., 2020).

The present supertagging approach differs from existing models in its training mechanism of word vectors, which are treated as intermediate products rather than end products. Category prediction is performed at the word, phrase, and sentence levels, inducing the training of vector representations of words that consider dependencies with other components.

Compute vectors for words/phrases, feed into a feed-forward neural network to form $P_w(i, i+1)$, $P_p(i, j)$ (category assignment probability distribution), and $P_s(i, j)$ (binary probability distribution of span existence), referring to Stern et al. (2017); Kitaev and Klein (2018) as

$$\begin{aligned}&P_w(i, i+1) \\ &= SM(\mathbf{Q}_w \sigma(LN(\mathbf{U}_w \mathbf{v}_{i:i+1} + \mathbf{b}_w) + \mathbf{c}_w)\end{aligned} \tag{11}$$

$$\begin{aligned}&P_p(i, j) \\ &= SM(\mathbf{Q}_p \sigma(LN(\mathbf{U}_p \mathbf{v}_{i:j} + \mathbf{b}_p)) + \mathbf{c}_p)\end{aligned} \tag{12}$$

$$\begin{aligned}&P_s(i, j) \\ &= SM(\mathbf{Q}_s \sigma(LN(\mathbf{U}_s \mathbf{v}_{i:j} + \mathbf{b}_s)) + \mathbf{c}_s),\end{aligned} \tag{13}$$

where $\mathbf{Q}_w, \mathbf{Q}_p, \mathbf{Q}_s, \mathbf{U}_w, \mathbf{U}_p, \mathbf{U}_s, \mathbf{b}_w, \mathbf{b}_p, \mathbf{b}_s, \mathbf{c}_w$ $\mathbf{c}_p$ and $\mathbf{c}_s$ denote the trainable parameters, $\sigma(\cdot)$ represents the nonlinear activation of the rectified linear unit (ReLU), and $LN(\cdot)$ indicates layer normalization, $SM(\cdot)$ denotes the softmax function. In addition, the dropout layer was immediately inserted after activation by the ReLU in each feedforward neural network.

Thereafter, we used backpropagation of multiple losses to train the model, using a corpus of CCG derivations and dependency structures

as the basis for losses. $\mathcal{L}_w$, $\mathcal{L}_p$, and $\mathcal{L}_s$ were computed based on cross-entropy loss between $P_w(i, i+1)$, $P_p(i, j)$, $P_s(i, j)$, and their supervised data $O_w(i, i+1)$, $O_p(i, j)$, and $O_s(i, j)$ (one-hot categorical distribution):

$$\mathcal{L}_w = -\sum_{i=0}^{n-1} \log(P_w(i, i+1))^\mathsf{T} O_w(i, i+1) \quad (14)$$

$$\mathcal{L}_p = -\sum_{(i,j)\in I_p} \log(P_p(i, j))^\mathsf{T} O_p(i, j) \quad (15)$$

$$\mathcal{L}_s = -\sum_{(i,j)\in I_s} \log(P_s(i, j))^\mathsf{T} O_s(i, j), \quad (16)$$

where log represents the element-wise logarithmic operation and I_p and I_s denote the set of span ranges in the training data. Thereafter, the model parameters were optimized by backpropagating a portion or all of the losses. In the case of backpropagation of all three losses, the calculation of the total loss and the update of the model parameters are expressed by the following equations:

$$\mathcal{L} = \mathcal{L}_w + \mathcal{L}_p + \mathcal{L}_s,\ \theta \leftarrow \theta - \mu \frac{\partial \mathcal{L}}{\partial \theta} \quad (17)$$

where $\mathcal{L}$ denotes the total loss to be backpropagated, θ represents the model parameters, and μ denotes the learning rate.

After training, the development and test data were supertagged by evaluating $P_w(i, i+1)$ and predicting the category assignment.

3.4 Parsing

In this section, we describe a method for incorporating phrase-level representations into span-based parsing, which searches for the binary tree maximizing the sum of log-likelihoods of category assignments and span existence, following the CKY algorithm. This framework is based on Stern et al. (2017) and Kitaev and Klein (2018).

Formulating CCG parsing, T was represented as a set of spans (i_t, j_t) with categories ℓ_t assigned.

$$T := \{(\ell_t, (i_t, j_t)) : t = 1, \ldots, |T|\}$$

Let $P_*(i, j)[\ell]$ and $P_s(i, j)[e]$ denote the probabilities of assigning a CCG category ℓ and the existence of span (i, j), respectively. The log-likelihood of the entire tree is computed by

$$\log P(T) = \sum_{(\ell,(i,j))\in T} [\log P_*(i, j)[\ell] + \log P_s(i, j)[e]], \quad (18)$$

and the problem of searching for the most plausible constituency tree $\hat{T}$ can be expressed as

$$\hat{T} = \underset{T}{\operatorname{argmax}} \log P(T), \quad (19)$$

where the subscript $*$ of $P_*(i, j)[\ell]$ represents w for $j = i + 1$; otherwise, p and $P_s(i, j)$ are defined using only $j \neq i + 1$.

In accordance with the presented formulation, Appendix A delineates our proposed span-based parsing method. The presence of unary rules within CCG's combinatory rules can potentially hinder not only the training process of the model but also the integration of a span-based parsing algorithm. While our assumption rests primarily on binary rules, unary rules—such as the transformation of N into NP—do exist. Consequently, this limits the proposed model's capability to delineate the procedure for vector composition.

In alignment with Stern et al. (2017), we consider the chain of categories processed by the unary rule as a unified category. We addressed this issue by transforming the CCG derivation into a form that could be represented by a complete binary tree; for instance, treating a chain of N to NP as a unified category N-NP based on the unary rule. This led to the prediction models of supertags and phrase types containing 1,340 and 948 category types, respectively.

Furthermore, inconsistencies may emerge in the categories of phrases and their constituent components (child nodes) based solely on the outcomes of category prediction. To address this issue, our proposed span-based parsing algorithm exclusively evaluates categories derived from child nodes, in compliance with the CCG combinatory rules, during the determination of the phrase category. This procedure is illustrated in lines 15 to 19 of Algorithm 1.

3.5 Decomposability

Our proposed model has a property allowing vector composition and decomposition, as expressed by Equations (20) and (21).

$$\mathbf{c} = \mathbf{a} \circ \mathbf{b} \quad (20)$$

$$\mathbf{b} = \mathbf{c} \diamond \mathbf{a} \quad (21)$$

where $\circ$ and $\diamond$ denote general composition and decomposition operations, respectively. In this formulation, decomposability is equivalent to automatically deriving $\mathbf{b}$ from $\mathbf{c}$ and $\mathbf{a}$.

In the proposed model, the composition operation is a circular correlation:

$$\mathbf{c} = \mathbf{a} \circ \mathbf{b} = \mathbf{a} \star \mathbf{b}. \quad (22)$$

Training Objectives	Norm Constraint	Parser	Acc	LF
$\mathcal{L}_w$ (baseline)	Real	C&C	96.41±0.03	91.77±0.03
$\mathcal{L}_w + \mathcal{L}_p$	Real	C&C	96.54±0.03	91.95±0.03
$\mathcal{L}_w + \mathcal{L}_s$	Real	C&C	96.54±0.03	91.94±0.04
$\mathcal{L}_w + \mathcal{L}_p + \mathcal{L}_s$	Real	C&C	**96.59±0.02**	92.03±0.04
		Span-based	-	**92.61±0.03**
	Complex	C&C	96.57±0.02	91.98±0.03
		Span-based	-	92.15±0.04

Table 1: Comparison of the supertagging accuracy (**Acc**) and labeled F-score (**LF**) on development data by training objective, norm constraint, and parser. In norm constraint, "Real" denotes normalization in real space, and "Complex" denotes complex unit magnitude projection. Both of these normalization methods are described in Section 3.2.

Assuming the complex unit magnitude projection of Section 3.2 is used for the vector's norm constraint, the decomposition operation can be derived by considering the inverse of Equation (5):

$$\mathbf{b} = \mathbf{c} \diamond \mathbf{a} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{c}) \oslash \overline{\mathcal{F}(\mathbf{a})}) \quad (23)$$

where $\oslash$ denotes the element-wise division. As discussed in Section 3.1, the circular correlation is a noncommutative operation, and if we need $\mathbf{a}$ instead of $\mathbf{b}$, then the decomposition operation needs to be modified as follows:

$$\mathbf{a} = \mathbf{c} \diamond \mathbf{b} = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{c})} \oslash \mathcal{F}(\mathbf{b})). \quad (24)$$

Here, CVG (Socher et al., 2013a) lacks decomposability due to the need for the matrix $W_{C \to AB}$ and lack of PCFG category information from the vectors themselves. In contrast, the current model has no such intervening parameters, enabling complete decomposition.

Vector decompositions enable text-infilling at phrase-level. Given vectors of words and phrases for input sentence *"My sister loves to eat"* (shown in Figure 3), we can reconstruct $\mathbf{v}_{0:2}$=*"My sister"* from $\mathbf{v}_{0:5}$ and $\mathbf{v}_{2:5}$=*"loves to eat"* as follows:

$$\mathbf{v}_{0:2} = \mathbf{v}_{0:5} \diamond \mathbf{v}_{2:5}. \quad (25)$$

Calculating the similarity between $\mathbf{v}_{0:2}$ and other vectors using cosine similarity enables retrieval of syntactically and semantically similar expressions. In this case, expected search results include phrases such as *"My brother"* and *"His sister"*.

This task does not necessarily require syntactic information and can be performed by mask prediction with large-scale language models (LLM). However, the prediction would be syntactically unnatural compared to our method, as shown in Tables 4 and 5. Additionally, the number of subwords to be predicted must be pre-determined when using LLM, thus precluding variable-length phrase filling as our method does. We tested the decomposability of our model in various cases on text-infilling tasks and compared the results to LLM.

4 Experiments

4.1 Datasets

We conducted experiments on CCGbank (Hockenmaier and Steedman, 2007), using a standard split scheme (02-21 section for training, 00 for development, and 23 for testing). Statistics on CCGbank are shown in Appendix B.

4.2 Training

We trained our model using different combinations of objectives, demonstrating the effectiveness of supertagging by training on only $\mathcal{L}_w$. We compared this baseline with those trained on $\mathcal{L}_w$, $\mathcal{L}_p$, and $\mathcal{L}_s$, thus enabling a single training process to satisfy both supertagging and parsing requirements.

To train $P_s(i, j)$, the actual spans in the gold derivation in CCGbank were treated as positive examples, and the spans generated by randomly and recursively splitting the span containing the entire sentence, but not included in gold derivation, were treated as negative examples.

We trained both the baseline model (minimized only $\mathcal{L}_w$) and the proposed model ($\mathcal{L}_p$ and $\mathcal{L}_s$ were subject to minimization) 10 times, with unique random seeds for each instance, and averaged the performance metrics for each model to perform a one-tailed t-test at the 1% significance level.

The proposed model contains 362 million trainable parameters, with 98.2% of these being derived from RoBERTa. In addition, we minimized the objectives using an AdamW optimizer (Loshchilov and Hutter, 2019). Model training takes around 2 hours using a single NVIDIA A100 GPU. Other hyperparameters are listed in Appendix C.

4.3 Parser Configuration

First, we conducted a parsing experiment using the Java version of the C&C parser (Clark et al., 2015) to demonstrate the effectiveness of our proposed supertagging model. We adopted a multitagging scheme, assigning supertags to each word with an assignment probability greater than 0.1, and used the default parameters of the C&C parser.

Model	Super-Tagger	Parser	Acc	LF
Lewis et al. (2016)	LSTM	A*	94.7	88.1
Vaswani et al. (2016)	LSTM	C&C	94.5	88.32
Yoshikawa et al. (2017)	LSTM	A* (LSTM)	–	88.8
Stanojević and Steedman (2020)	LSTM	Shift-Reduce (LSTM)	–	90.6
Tian et al. (2020)	Attentive-GCNN	EasyCCG	96.25	90.58
Bhargava and Penn (2020)	LSTM decoder	C&C	96.00	90.9
Liu et al. (2021)	Category Generator	C&C	96.05	90.87
Prange et al. (2021)	Tree-Structured decoder	C&C	96.22	90.91
Kogkalidis and Moortgat (2022)	Heterogeneous Dynamic Convolutions	–	96.29	–
Clark (2021)	Tian et al. (2020)	C&C	–	91.9
		Span-based	–	**92.9**
Ours ($\mathcal{L}_w + \mathcal{L}_p + \mathcal{L}_s$, Real)	Holographic	C&C	**96.60**	**92.12**
		Span-based	–	92.67

Table 2: Comparison of the proposed model and existing methods; best results are shown in bold.

Operator	Acc	LF
Corr ($\star$)	96.59±0.02	92.61±0.03
Conv ($*$)	96.57±0.02	92.75±0.02
s-Conv ($\otimes$)	96.54±0.02	92.12±0.04

Table 3: Performance comparison on development data using different compositional operators, measured by accuracy for supertagging and labeled F-score for parsing. Corr, Conv, and s-Conv denote circular correlation, circular convolution, and shuffled circular convolution, respectively.

Subsequently, we conducted an experiment with our proposed span-based parsing algorithm to demonstrate phrase-level category assignment influence on parsing. For evaluation, we extracted dependencies from CCG derivation using the `generate` program of the C&C parser. Variations in grammatical constraints caused programmatic extraction failure for some sentences, so their dependencies were replaced by C&C parser results.

Furthermore, we implemented the `skimmer` mode for our span-based parsing algorithm, along with the C&C parser, enabling the detection of dependencies between words, even if the parser is unable to parse the entire sentence. Consequently, our parser achieved 100% coverage.

5 Results

5.1 Supertagging Accuracy

Table 1 presents supertagging accuracy on development data for each training loss combination. First, we compared models with norm constraints on real space and found our proposed models to be statistically superior to the baseline in terms of supertagging accuracy. Moreover, the supertagging performance varied slightly compared to the model with norm constraints on the complex space, indicating a low impact of the type of norm constraint.

Table 2 shows the proposed supertagging model outperforming existing models, achieving a new state of the art. This indicates the effectiveness of the proposed approach in inducing category assignments at the word level while considering phrase-level representations.

5.2 Parsing Performance

The labeled F-scores of the current span-based parser and C&C parser on the development data are presented in Table 1. First, the model with norm constraints on the real space outperformed the baseline, even with the same C&C parser, due to improved supertagging. Furthermore, compared with C&C, the proposed span-based parsing algorithm improved performance for the model with norm constraint on real space. However, the performance gap between C&C and the model with norm constraints on complex space is relatively small. This implies that models' expressive power with the norm constraint on complex space is limited compared to real space. This could be due to representations being distributed on a d-dimensional unit hypersphere in complex space, thus lacking norm information along each dimension.

Model using proposed supertagging approach and C&C outperformed all existing models with the same parser (Table 2). Furthermore, the performance of the proposed span-based parsing model is comparable to that of the current state-of-the-art model of Clark (2021) using Transformers. Overall, results indicate recursive holographic compositions improve CCG parsing performance.

5.3 Replacement of Compositional Operator

Examining performance gaps when employing alternative compositional operators in our method (Table 3) revealed little difference in performance for supertagging. However, the application of shuffled circular convolution exhibited lower performance than the other two operators for parsing.

ID	Sentence	Replacement by Holographic CCG	Sim.	NPMI
1	*Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group .*	*Mr. Baris*	1.00	0.19
		Dr. Novello	1.00	0.10
		Ms. Ensrud	1.00	0.11
2	*When Scoring High first came out in 1979 , it was a publication of Random House .*	*turned up*	0.94	0.27
		sold out	0.91	0.29
		sells out	0.90	0.24
3	*In early trading in Hong Kong Thursday , gold was quoted at $ 374.19 an ounce .*	*for $ 25.50 a share*	0.94	0.33
		for $ 60 a bottle	0.94	0.29
		at $ 51.25 a share	0.93	0.34
4	*Judges are not getting what they deserve .*	*what she did*	0.96	0.28
		what they do	0.96	0.36
		what we do	0.89	0.35
5	*Despite recent declines in yields , investors continue to pour cash into money funds .*	*Despite the flap over transplants*	0.89	0.22
		In a victory for environmentalists	0.86	0.22
		On the issue of abortion	0.82	0.27
6	*Despite recent declines in yields , investors continue to pour cash into money funds .*	*to provide maintenance for other manufacturers*	0.83	0.27
		to share data via the telephone	0.79	0.21
		to cut costs throughout the organization	0.77	0.26

Table 4: List of target sentences, phrases, and candidates for replacement. The underlined parts of the sentence denote phrases for reconstruction and replacement and **Sim.** indicates the cosine similarity between the reconstructed vector and the replacement candidate vector. **NPMI** shows the mean of the values calculated for each word among the replacement candidates.

ID	Replacement by RoBERTa	NPMI
1	*A.P. Bates*	0.11
	Ms. Vinken	0.35
	Dyearella Sr.	0.08
2	*was introduced*	0.32
	went open	0.22
	took place	0.23
3	*with $ 368.24 an ounce*	0.38
	as $ 368.79 an ounce	0.38
	at $ 368.24 a piece	0.31
4	*difficult to defend*	0.23 †
	at their views	0.26 †
	out of themselves	0.28 †
5	*To provide a defensive edge*	0.26 †
	In a routine shakeup	0.20
	After several years of weakness	0.26
6	*on a trend toward lower yields*	0.32 †
	6 ignore the quake in California	0.24
	getting scared out of their lives	0.21

Table 5: List of replacement candidates and NPMI with mask prediction using RoBERTa. IDs are consistent with those of Table 4. Replacement candidates with † mean that the outermost non-terminal symbol given by Berkeley Neural Parser is different from that of the original phrase in the sentence.

This difference may be attributed to the presence of a permutation matrix Φ in Equation (8), unlike the other two. As for parsing, circular convolution was slightly superior to circular correlation, yet the small performance gap is not considered serious enough to preclude circular correlation in our approach, due to its desirable properties for embedding phrase structures and potential for composing semantic and syntactic information.

5.4 Decomposition

We present a qualitative evaluation of text-infilling, enabled by the decomposability of our model. We reconstruct phrase vectors from development data and compare them to the vectors of phrases in other sentences to select the top-n most similar phrases as candidate replacements, following Tian et al. (2016). Then we compare our proposed decomposition method (Table 4) with fine-tuned RoBERTa for mask prediction (Table 5).

Results indicate our method found expressions more syntactically similar to the original. E.g. for ID 4, our method output all relative pronoun phrases beginning with *"what"*, and for ID 6, infinitive phrases starting with *"to"*, as did the original expression, unlike RoBERTa. In addition, we used Berkeley Neural Parser (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2018) for the analysis of the original and replaced sentences and showed that all the non-terminal symbols in the pre- and post-replaced phrases matched in our method, whereas different non-terminal symbols were assigned in RoBERTa in some cases (syntactic structure has changed). Randomly selecting phrases of length 2-6 from sentences of length 10-30 (total 1,285 sentences) in development data, our method achieved a 96.31% match rate of outermost nonterminals, compared to 77.95% for RoBERTa.

Furthermore, we calculated normalized pointwise mutual information (NPMI; Bouma 2009) to evaluate semantic naturalness; a two-tailed t-test

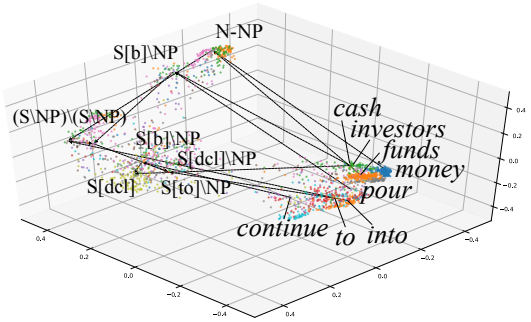

Figure 4: Recursive vector composition on embedding space for a sentence *"investors continue to pour cash into money funds"*, visualized by PCA. The color of each dot represents a unique CCG category. CCG categories for words are as follows: *money*: `N/N`, *funds*: `N`, *cash*, *investors*: `N-NP`, *continue*: `(S[dcl]\NP)/(S[to]\NP)`, *to*: `(S[to]\NP)/(S[b]\NP)`, *pour*: `(S[b]\NP)/ NP`, *into*: `((S\NP)\(S\NP))/NP`.

showed no significant difference in mean NPMI of the proposed method and RoBERTa (0.255 vs. 0.258; p-value = 0.910). Our model can provide syntactically and semantically natural replacement, despite focusing on syntactic information.

6 Conclusion

In this paper, we proposed a novel method for formulating CCG as a recursive composition operation on a continuous vector space and constructing phrase/sentence-level representations from word representations. We demonstrated its utility for supertagging and parsing. Experimentation demonstrated the effectiveness of holographic compositions in explicitly modelling dependencies between sentence components, resulting in improved performance and state-of-the-art results in supertagging and parsing using the C&C parser. In addition, we validated that phrase-level text-infilling is possible by applying the decomposable property of the holographic representation in the proposed model.

7 Limitations

Firstly, the training process of our proposed model is dependent on supervised data, thus precluding its application to languages without a supervised dataset for CCG.

Also, the span-based parsing algorithm proposed in this study is implemented in Python and may take a considerable amount of time to parse extremely long sentences (more than 100 words) due to a lack of optimization for implementation.

Acknowledgements

This work was supported by JST, Moonshot R&D Grant Number JPMJMS2033, and JSPS KAKENHI Grant Number JP23H04835.

References

Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. 2020. BoxE: A box embedding model for knowledge base completion. In *Advances in Neural Information Processing Systems*, volume 33, pages 9649–9661. Curran Associates, Inc.

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.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1240–1246, Geneva, Switzerland. COLING.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, 30:31–40.

Stephen Clark. 2021. Something old, something new: Grammar-based CCG parsing with transformer models. *CoRR*, abs/2109.10044v2.

Stephen Clark and James R. Curran. 2007. Wide-coverage 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, August*.

Ashwinkumar Ganesan, Hang Gao, Sunil Gandhi, Edward Raff, Tim Oates, James Holt, and Mark McLean. 2021. Learning with holographic reduced representations. In *Advances in Neural Information Processing Systems*.

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.

Nikita Kitaev, Steven Cao, and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. In *Annual Meeting of the 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.

Konstantinos Kogkalidis and Michael Moortgat. 2022. Geometry-aware supertagging with heterogeneous dynamic convolutions. *CoRR*, abs/2203.12235v2.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692v1.

Yufang Liu, Tao Ji, Yuanbin Wu, and Man Lan. 2021. Generating CCG categories. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13443–13451.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Pascual Martinez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1955–1961. AAAI Press.

T.A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2015. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

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

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013b. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 129–136, Madison, WI, USA. Omnipress.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013c. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. 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*. MIT Press, Cambridge, MA, USA.

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.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional

distributional representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1277–1287, Berlin, Germany. 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.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org.

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.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. 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.

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 115–122, Madison, WI, USA. Omnipress.

Fabio Massimo Zanzotto, Andrea Santilli, Leonardo Ranaldi, Dario Onorati, Pierfrancesco Tommasino, and Francesca Fallucchi. 2020a. KERMIT: Complementing transformer architectures with encoders of explicit syntactic interpretations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 256–267, Online. Association for Computational Linguistics.

Fabio Massimo Zanzotto, Giorgio Satta, and Giordano Cristini. 2020b. CYK parsing over distributed representations. *Algorithms*, 13(10).

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320, San Diego, California. Association for Computational Linguistics.

Yanpeng Zhao, Liwen Zhang, and Kewei Tu. 2018. Gaussian mixture latent vector grammars. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1189, Melbourne, Australia. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1604–1612, Lille, France. PMLR.

A Span-based Parsing Algorithm

Our proposed novel span-based parsing algorithm is shown in Algorithm 1. Although the basic flow of the algorithm remained the same as that of the original CKY algorithm, there were certain modifications related to the incorporation of phrase-level representations which are explained in detail by associating line numbers in the Algorithm 1.

First, in line 1, the input word sequence $(w_1, w_2, \cdots, w_n)$ was converted into vectors $(\mathbf{v}_{0:1}, \mathbf{v}_{1:2}, \cdots, \mathbf{v}_{n-1:n})$ using the encoder, and in lines 2 to 6, the categories with a higher probability of assignment to each word were stored in the chart along with the log-likelihood of their assignment. In particular, the unique feature of this algorithm pertains to line 6, wherein the vector of each word is stored in a separate chart to compute the vector of phrases at a later stage. Moreover, the vector $\mathbf{v}_{i:j}$ of the span (i, j) to be split at split point k using circular correlation is stated in line 16, and this vector is used for calculating the probability distribution of span existence and category assignment to the phrases in line 17 and 19. After conducting the two-step thresholding process in lines 18

Algorithm 1: Span-based CKY parsing

1 $\mathbf{v}_{0:1}, \mathbf{v}_{1:2}, \cdots, \mathbf{v}_{n-1:n} = Encode(w_1, w_2, \cdots, w_n)$;
2 **for** $i = 0, \cdots, n-1$ **do**
3 $P_w(i, i+1) = SM(\mathbf{Q}_w \sigma(LN(\mathbf{U}_w \mathbf{v}_{i:i+1} + \mathbf{b}_w)) + \mathbf{c}_w)$; ▷ Equation (11)
4 **for** $C \in \{X | P_w(i, i+1)[X] > t_w = 0.1\}$ **do**
5 $prob[i, i+1, C] = \log P_w(i, i+1)[C]$;
6 $vector[i, i+1, C] = \mathbf{v}_{i:i+1}$;
7 **for** $\ell = 2, \cdots, n$ **do**
8 **for** $i = 0, \cdots, n-\ell$ **do**
9 $j = i + \ell$;
10 **for** $k = i+1, \cdots, j-1$ **do**
11 **for** $C_1 \in \{X | prob[i, k, X] > 0\}$ **do**
12 $\mathbf{v}_{i:k} = vector[i, k, C_1]$;
13 **for** $C_2 \in \{X | prob[k, j, X] > 0\}$ **do**
14 $\mathbf{v}_{k:j} = vector[k, j, C_2]$;
15 **for** $C \in \{X | C_1 C_2 \rightarrow X \in R\}$ **do**
16 $\mathbf{v}_{i:j} = \mathbf{v}_{i:k} \star \mathbf{v}_{k:j}$; ▷ Equations (4) and (5)
17 $P_s(i, j) = SM(\mathbf{Q}_s \sigma(LN(\mathbf{U}_s \mathbf{v}_{i:j} + \mathbf{b}_s)) + \mathbf{c}_s)$; ▷ Equation (13)
18 **if** $P_s(i, j)[e] > t_s = 0.01$ **then**
19 $P_p(i, j) = SM(\mathbf{Q}_p \sigma(LN(\mathbf{U}_p \mathbf{v}_{i:j} + \mathbf{b}_p)) + \mathbf{c}_p)$; ▷ Equation (12)
20 **if** $P_p(i, j)[C] > t_p = 0.01$ **then**
21 $p = \log P_p(i, j)[C] + \log P_s(i, j)[e] + prob[i, k, C_1] + prob[k, j, C_2]$;
22 **if** $p > prob[i, j, C]$ **then**
23 $prob[i, j, C] = p$;
24 $backpointer[i, j, C] = (k, C_1, C_2)$;
25 $vector[i, j, C] = \mathbf{v}_{i:j}$;

and 20, the log-likelihood of assigning category C, which was combined from categories C_1 and C_2 following the combinatory rule R, to span (i, j) was calculated in line 21 based on Equation (18).

In implementing the combinatory rules used in the algorithm (R in line 15), we employed all combinatory rules that appeared at least once in the training data. This allows for a larger search space and simpler program implementation compared to existing methods.

B CCGbank Statistics

Table 6 presents the statistics of CCGbank which we used for our experiments.

C Hyperparameters

Table 7 presents the list of hyperparameters used in our experiments.

	Train	Dev	Test
Section number	02-21	00	23
Number of sentences	39,604	1,913	2,407
Number of words	929,552	45,422	55,371

Table 6: Statistics of CCGbank.

Hyperparameters	**Values**
k in Equation (9)	30
ϵ in Equation (9)	1e-12
Training epochs	10
Batch size	16
Learning rates	1e-4(base), 1e-5(fine-tune)
AdamW $\beta's$	0.9, 0.999
AdamW ϵ	1e-6
Weight decay	0.01
Dropout probability	0.2

Table 7: List of hyperparameters used in our experiments. Among the model components, we adopted various learning rates for the encoding component using RoBERTa-large (fine-tune) and the other component (base).