Supertagging Combinatory Categorial Grammar with Attentive Graph Convolutional Networks

Yuanhe Tian♥, **Yan Song**♠♡†, **Fei Xia**♥
♥University of Washington ♠The Chinese University of Hong Kong (Shenzhen)
♡Shenzhen Research Institute of Big Data
♥{yhtian, fxia}@uw.edu ♠songyan@cuhk.edu.cn

Abstract

Supertagging is conventionally regarded as an important task for combinatory categorial grammar (CCG) parsing, where effective modeling of contextual information is highly important to this task. However, existing studies have made limited efforts to leverage contextual features except for applying powerful encoders (e.g., bi-LSTM). In this paper, we propose attentive graph convolutional networks to enhance neural CCG supertagging through a novel solution of leveraging contextual information. Specifically, we build the graph from chunks (n-grams) extracted from a lexicon and apply attention over the graph, so that different word pairs from the contexts within and across chunks are weighted in the model and facilitate the supertagging accordingly. The experiments performed on the CCGbank demonstrate that our approach outperforms all previous studies in terms of both supertagging and parsing. Further analyses illustrate the effectiveness of each component in our approach to discriminatively learn from word pairs to enhance CCG supertagging.[1]

1 Introduction

Combinatory categorial grammar (CCG) is a lexicalized grammatical formalism, where the lexical categories (also known as supertags) of the words in a sentence provide informative syntactic and semantic knowledge for text understanding. Therefore, CCG parse often provides useful information for many downstream natural language processing (NLP) tasks such as logical reasoning (Yoshikawa et al., 2018) and semantic parsing (Beschke, 2019). To perform CCG parsing in different languages, most studies conducted a supertagging-parsing pipline (Clark and Curran, 2007; Kummerfeld et al., 2010; Song et al., 2012; Lewis and Steedman, 2014b; Huang and Song, 2015; Xu et al., 2015; Lewis et al., 2016; Vaswani et al., 2016; Yoshikawa et al., 2017), in which their main focus is the first step, and they generated the CCG parse trees directly from supertags with a few rules afterwards.

Building an accurate supertagger in a sequence labeling process requires a good modeling of contextual information. Recent neural approaches to supertagging mainly focused on leveraging powerful encoders with recurrent models (Lewis et al., 2016; Vaswani et al., 2016; Clark et al., 2018), with limited attention paid to modeling extra contextual features such as word pairs with strong relations. Graph convolutional networks (GCN) is demonstrated to be an effective approach to model such contextual information between words in many NLP tasks (Marcheggiani and Titov, 2017; Huang and Carley, 2019; De Cao et al., 2019; Huang et al., 2019); thus we want to determine whether this approach can also help CCG supertagging.

However, we cannot directly apply conventional GCN models to CCG supertagging because in most of the previous studies the GCN models are built over the edges in the dependency tree of an input sentence. As high-quality dependency parsers are not always available, we do not want our CCG supertaggers to rely on the existence of dependency parsers. Thus, we need another way to extract useful word pairs to build GCN models. For that, we propose to obtain word pairs from frequent chunks (n-grams) in the corpus, because those chunks are easy to identify with co-occurrence counts. To appropriately learn from n-grams, one requires the GCN to be able to distinguish different word pairs because such information in n-grams are not explicitly structured as that in dependency parses. Because existing GCN models are limited in treating all word pairs equally, we propose an adaptation of conventional GCN for CCG supertagging.

[†]Corresponding author.

[1]Our code and models for CCG supertagging are released at https://github.com/cuhksz-nlp/NeST-CCG.

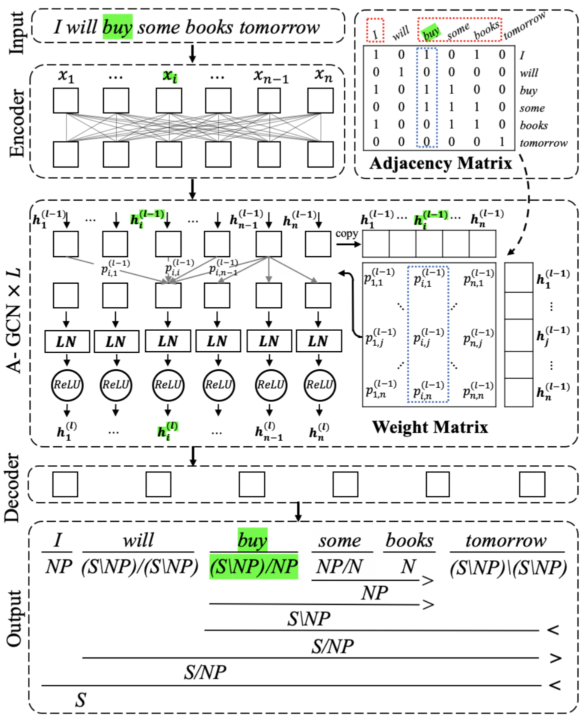

Input
I will buy some books tomorrow
Encoder
Adjacency Matrix
A- GCN × L
LN
ReLU
copy
Weight Matrix
Decoder
Output
I will buy some books tomorrow
NP (S\NP)/(S\NP) (S\NP)/NP NP/N N (S\NP)\(S\NP)
NP
S\NP
S/NP
S/NP
S

Figure 1: The architecture of our CCG supertagger with A-GCN and an example input sentence with its supertagging and parsing output. The supertagging process for "*buy*" is highlighted in green. The adjacency matrix illustrates the edges of the graph that is built upon the chunks (n-grams) extracted from the lexicon $\mathcal{N}$, with the chunks illustrated in the red boxes.

In this paper, we propose attentive GCN (A-GCN) for CCG supertagging, where its input graph is built based on chunks (n-grams) extracted with unsupervised methods. In detail, two types of edges in the graph are introduced to model word relations within and across chunks and an attention mechanism is applied to GCN to weight those edges. In doing so, different contextual information are discriminatively learned to facilitate CCG supertagging without requiring any external resources. The validity of our approach is demonstrated by experimental results on the CCGbank (Hockenmaier and Steedman, 2007), where state-of-the-art performance is obtained for both tagging and parsing.

2 The Approach

We treat CCG supertagging as a sequence labeling task, where the input is a sentence with n words $\mathcal{X} = x_1 x_2 \cdots x_i \cdots x_n$, and the output is a sequence of supertags $\widehat{\mathcal{Y}} = \widehat{y}_1 \widehat{y}_2 \cdots \widehat{y}_i \cdots \widehat{y}_n$. Our approach uses attentive GCN (A-GCN) to incorporate information of word pairs through a graph; the graph is built based on n-grams in the input sentence that appear in a lexicon $\mathcal{N}$. This lexicon consists of n-grams automatically extracted from raw corpora by unsupervised methods. The overall architecture of our tagger is illustrated in Figure 1, with an input sentence and corresponding supertagging and parsing output. The details of the main components in the architecture are provided below.

2.1 GCN

Normal GCN models with L layers learn from word pairs suggested by the dependency parsing results of the input sentence $\mathcal{X}$, where the edges between all pairs of words x_i and x_j are represented by an adjacency matrix $\mathcal{A} = \{a_{i,j}\}_{n\times n}$. In $\mathcal{A}$, $a_{i,j} = 1$ if there is a dependency edge between x_i and x_j or $i = j$ (the direction of the edge is ignored), and $a_{i,j} = 0$ otherwise. Based on the adjacency matrix, for each x_i, the l-th GCN layer finds all x_j associated with x_i (where $a_{i,j} = 1$), takes their hidden vectors $\mathbf{h}_j^{(l-1)}$ from the $(l-1)$-th layer, and computes the output for x_i by

$$\mathbf{h}_i^{(l)} = \sigma(LN(\sum_{j=1}^{n} a_{i,j}(\mathbf{W}^{(l)} \cdot \mathbf{h}_j^{(l-1)} + \mathbf{b}^{(l)}))) \quad (1)$$

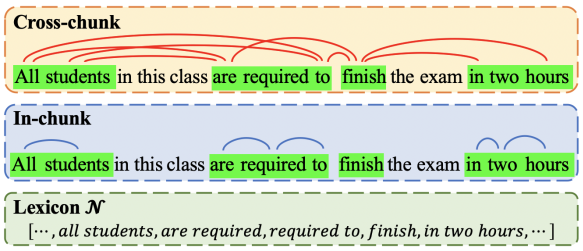
where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are trainable matrix and bias for the l-th GCN layer, LN refers to layer normalization and σ the $ReLU$ activation function. Therefore, in normal GCN, for each x_i, all the x_j that connect to x_i are treated exactly the same.

2.2 Graph Construction based on Chunks

Since CCG supertagging is also a parsing task, we do not want our approach to rely on the existence of a dependency parser. Without such a parser, we need an alternative for finding good word pairs to build the graph in A-GCN (which is equivalent to build the adjacency matrix $\mathcal{A}$). Inspired by the studies that leverage chunks (n-grams) as effective features to carry contextual information and enhance model performance (Song et al., 2009; Song and Xia, 2012; Ishiwatari et al., 2017; Yoon et al., 2018; Zhang et al., 2019; Tian et al., 2020a,c,b), we propose to construct the graph based on the chunks (n-grams) extracted from a pre-constructed n-gram lexicon $\mathcal{N}$. Specifically, the lexicon is constructed by computing the PMI of any two adjacent words s', s'' in the training set by

$$PMI(s', s'') = log\frac{p(s's'')}{p(s')p(s'')} \quad (2)$$

where p is the probability of an n-gram (i.e., s', s'' and $s's''$) in the training set; then a high PMI


Cross-chunk
All students in this class are required to finish the exam in two hours
In-chunk
All students in this class are required to finish the exam in two hours
Lexicon $\mathcal{N}$
[$\cdots$, all students, are required, required to, finish, in two hours, $\cdots$]
Figure 2: Examples of the two types of edges for building the graph in an input sentence, in which chunks (n-grams) extracted from the lexicon $\mathcal{N}$ are highlighted in green; example in-chunk and cross-chunk edges are marked in blue and red color, respectively.

score suggests that the two words co-occur a lot in the dataset and are more likely to form a n-gram. For each pair of adjacent words s_{i-1}, s_i in a sentence $\mathcal{S} = s_1 s_2 \cdots s_{i-1} s_i \cdots s_n$, we compute the PMI score of the two words and use a threshold to determine whether a delimiter should be inserted between them. As a result, the sentence $\mathcal{S}$ is segmented into pieces of n-grams and we extract all n-grams from all sentences to form the lexicon $\mathcal{N}$.[2]

Then for graph building, given an input sentence $\mathcal{X}$, we find all the n-grams in $\mathcal{X}$ that appear in $\mathcal{N}$. A *chunk* is either a n-gram that does not overlap with other n-grams or a text span that covers multiple overlapping n-grams. For example, in Figure 2, we find four chunks (i.e., "*all students*", "*are required to*", "*finish*", and "*in two hours*") in the example sentence according to the lexicon $\mathcal{N}$ (the chunks are highlighted in green). In these chunks, "*all students*", "*finish*", and "*in two hours*" are non-overlapping n-grams included in the lexicon and "*are required to*" is a text span that covers the overlapping n-grams "*are required*" and "*required to*". In most cases, the adjacent words within the same chunk tend to have a strong word-word relation in terms of co-occurrence, and thus we can build the graph and its adjacency matrix accordingly.

Based on the chunks, we construct the graph by two types of edges, i.e., the in-chunk and cross-chunk ones: the first type is to model local word pairs, and the graph includes edges between any two adjacent words within the same chunk. For example, as shown in Figure 2, the in-chunk edges (blue lines) for the chunk "*in two hours*" are "(*in, two*)" and "(*two hours*)". The second type is to model cross chunk word pairs, which are built from any two adjacent chunks with the starting and ending words in the two chunks connected. The motivation of using the starting and ending words is that English phrases tend to be head-initial (e.g., verb phrase such as "*buy some books*") or head-final (e.g., adjective phrase such as "*red apples*") in many cases. E.g., for the two chunks "*all students*" and "*are required to*" in Figure 2, the corresponding cross-chunk edges (red lines) are "*(all, are)*", "*(all, to)*", "*(students, are)*", and "*(students, to)*". The graph is equivalent to the adjacency matrix $\mathcal{A}$, where $a_{i,j} = 1$ if there is an edge between x_i and x_j in the graph or $i = j$, and $a_{i,j} = 0$ otherwise.[3]

2.3 The Attentive GCN

When learning from a graph, conventional GCN models treat all word pairs from the graph equally, and thus are unable to account for the possibility that the contribution of different x_j on x_i may vary. Particularly for our graph built from chunks, it is important to be able to distinguish different word pairs because all the chunks and the graph are constructed automatically without a dependency parser. Therefore, we apply an attention mechanism to the adjacency matrix and adapt Eq. (1) used in the normal GCN for our A-GCN by replacing the $a_{i,j} \in \{0, 1\}$ by a weight $p^{(l)}_{i,j} \in (0, 1)$. For each x_i and all its associated x_j, the weight $p^{(l)}_{i,j}$ for this word pair is computed by

$$p^{(l)}_{i,j} = \frac{a_{i,j} \cdot exp(\mathbf{h}^{(l-1)}_i \cdot \mathbf{W}^{(l)}_{pos} \cdot \mathbf{h}^{(l-1)}_j)}{\sum_{j=1}^{n} a_{i,j} \cdot exp(\mathbf{h}^{(l-1)}_i \cdot \mathbf{W}^{(l)}_{pos} \cdot \mathbf{h}^{(l-1)}_j)} \tag{3}$$

where $\mathbf{W}^{(l)}_{pos}$ models the positional relation (i.e., *left*, *right*, or *self*) between x_i and x_j and it has three choices, i.e., $\mathbf{W}^{(l)}_{left}$, $\mathbf{W}^{(l)}_{right}$, and $\mathbf{W}^{(l)}_{self}$ for different i and j,[4] with each of them a trainable square matrix in the l-th layer of A-GCN.

2.4 Supertagging with A-GCN

To conduct supertagging with A-GCN, we firstly obtain the hidden vector $\mathbf{h}^{(0)}_i$ for x_i from BERT (Devlin et al., 2019) to feed into the first GCN layer. Upon receiving the encoding results from A-GCN, the following supertagging process is straightforward: each $\mathbf{h}^{(L)}_i$ is obtained from the last A-GCN layer and aligned with the output by $\mathbf{o}_i = \mathbf{W}_d \cdot \mathbf{h}^{(L)}_i$, where $\mathbf{W}_d$ is a trainable matrix

[2]For example, a sentence can be segmented into $\mathcal{S} = s_1/s_2 s_3 s_4/s_5$ ("/" refers to a delimiter) if the PMI of s_1, s_2 and s_4, s_5 are lower than the threshold and the PMI of s_2, s_3 and s_3, s_4 are greater than the threshold; we thus obtain three n-grams, i.e., s_1, $s_2 s_3 s_4$, and s_5 from this sentence.

[3]We do not distinguish the two types of edges in $\mathcal{A}$.

[4]For example, $\mathbf{W}^{(l)}_{pos} = \mathbf{W}^{(l)}_{left}$, if $j < i$.

	Train	Dev	Test
Section No.	2-21	0	23
Sent #	39,604	1,913	2,407
Word #	44,210	7,878	8,392

Table 1: The train/dev/test splits of English CCGBank and the statistics of sentences and words in them.

Hyper-parameters	**Values**
Batch Size	16, 32
Drop-out Rate	0.2
Learning Rate	3e-5, 2e-5, 1e-5, 5e-6
Max Sentence Length	300
Random Seed	42
Training Epoch	50
Warm-up Rate	0.1, 0.2

Table 2: The list of hyper-parameters tested in our experiments. We run all models with the combination of those hyper-parameters and use the one achieving the highest supertagging results in our final experiments.

for the alignment. Then, a *softmax* decoder is used to predict the supertag $\hat{y}_i$ for x_i:

$$\hat{y}_i = \arg\max \frac{exp(\mathbf{o}_i^t)}{\sum_{t=1}^{|\mathcal{T}|} exp(\mathbf{o}_i^t)} \tag{4}$$

where $\mathcal{T}$ denotes the set with all CCG categories and o_i^t the value at dimension t in $\mathbf{o}_i$.

3 Experiments

3.1 Settings

We run experiments on the English CCGbank (Hockenmaier and Steedman, 2007)[5] and follow Clark and Curran (2007) to split it into train/dev/test sets, whose statistics (sentence and word numbers) are reported in Table 1. To construct n-gram lexicon $\mathcal{N}$ for building the edges in our graph, we perform PMI on the training set of CCGbank to extract n-grams whose length is between $[1, 5]$, with the threshold of the PMI score set to 0. For the encoder, we try both cased and uncased BERT-Large (Devlin et al., 2019) with their default settings (e.g., 24 layers of self-attentions in 1024 dimensional hidden vectors)[6] and used two layers for A-GCN. To obtain CCG parse from the generated supertags, we adopt the parsing algorithm used in EasyCCG (Lewis and Steedman, 2014a). We follow previous studies (Lewis and Steedman, 2014a; Lewis et al., 2016; Yoshikawa et al., 2017) to evaluate our model on both the tagging accuracy of the most frequent 425 supertags and the labeled F-scores (LF) of the dependencies converted from CCG parse[7].

For other hyper-parameter settings, we test their values as shown in Table 2 when training our models. We tried all combinations of them for each model and use the one achieving the highest supertagging results in our final experiments. Note that, with the best hyper-parameters, the best performance is achieved with warm-up rate 0.1, batch size 16, and learning rate 1e-5.

Models	**PARM**	**TAG**	**LF**
BERT-Cased	335M	96.09	90.25
+ A-GCN (Full)	343M	95.98	90.03
+ A-GCN (Chunk)	343M	**96.24**	**90.42**
BERT-Uncased	337M	96.14	90.32
+ A-GCN (Full)	345M	96.08	90.13
+ A-GCN (Chunk)	345M	**96.31**	**90.52**

Table 3: Results (supertagging accuracy and labeled F-scores) of different models with BERT-Large encoder on the development set of CCGbank. "PARM" is the number of trainable parameters in the models; "Full" uses the fully connected graph and "Chunk" uses the graph built based on chunks.

3.2 Results

To explore the effectiveness of our approach, we run CCG taggers with and without A-GCN, and try two ways to construct the graph: one is a fully connected GCN where edges are built between every two words; the other is our proposed approach with the chunk-based graph. Experimental results on supertagging accuracy (TAG) and labeled F-scores (LF) for parsing on the development set of CCGbank are reported in Table 3, with the number of trainable parameters of all models also presented.

The experiments show that, for both cased and uncased BERT encoders, the proposed chunk A-GCN works the best in terms of both supertagging accuracy and parsing results. In contrast, Full A-GCN has inferior performance to the BERT baselines. This contrast shows the importance of appropriate construction of the graphs fed into A-GCN, since the fully connected graph with all words asso-

[5]The official dataset is obtained from https://catalog.ldc.upenn.edu/LDC2005T13.

[6]We download the pre-trained BERT models from https://github.com/google-research/bert.

[7]We use the "*generate*" script from C&C parser (Clark and Curran, 2007) to convert CCG parse results to their corresponding dependencies.

Models	TAG	LF
Lewis and Steedman (2014b)	91.3	86.11
Xu et al. (2015)	93.00	87.07
Lewis et al. (2016)	94.7	88.1
Vaswani et al. (2016)	94.24	88.32
Yoshikawa et al. (2017)	-	90.4
Clark et al. (2018)	96.1	-
Stanojević and Steedman (2019)	95.4	90.5
EasyCCG†	-	86.14
BERT†	96.25	90.46
BERT + A-GCN (Full)†	96.10	90.34
BERT + A-GCN (Chunk)†	**96.39**	**90.68**

Table 4: Comparison of our models with uncased BERT encoder and previous studies on the test set of CCGbank. Models with "†" use the EasyCCG parser to generate CCG parse trees from the predicted supertags.

ciated with one another may introduce noise word relations and thus yield bad performance.

Furthermore, we run our models with uncased BERT encoder on the test set and compare the performance with previous studies on both supertagging and parsing. Table 4 shows the results, where the studies marked by † use the same parser (i.e., the EasyCCG parser) to generate CCG trees from supertags. Among the previous studies, Stanojević and Steedman (2019) performed CCG parsing directly without the suppertagging step, whereas the rest all did supertagging first. Regardless of this difference, our approach performs the best on CCGbank in both supertagging accuracy and parsing LF.

3.3 Ablation Study

We conduct an ablation study to explore the effect of the two types of edges and the attention mechanism on our best model. The supertagging and parsing results of models with different configurations are reported in Table 5, where the results are categorized into four groups. The first group (ID 1) is the results of the best performing model where all settings are activated; the second (ID 2-3) is the ablation of either in-chunk or cross-chunk edges with attention; the third (ID 4-6) is the result of using normal GCN without the attention mechanism; and the last group (ID 7) is the baseline model where none of the three settings is activated.

The results show that the model performance drops when either part is ablated (ID 1 vs. ID 2-6). Specifically, removing attention significantly hurts the performance, where all configurations without attention (ID 4-6) shows worse-than-baseline (ID 7) results; this observation confirms the importance of applying attention on GCN. One possible explanation to this phenomenon could be that considerable noises are introduced to the graph because the edges in our graph are derived from chunks and they do not follow syntax in most cases; thus, it is crucial to assign weights to the edges and not treat them with equally. Interestingly, comparing the two types of edges, models with cross-chunk edges yield much higher results than the ones with in-chunk edges only when the attention is not used (ID 5 vs. ID 6), while it is slightly better when attention is applied (ID 2 vs. ID 3). This comparison suggests that in-chunk edges could introduce more noise than cross-chunk edges. So that when the attention is not used (ID 6), the model fails to weight the edges and results in a significant drop on its performance; On the contrary, when the attention is applied (ID 3), our model is able to even the performance of models with in-chunk and cross-chunk edges, which confirms that weighting is essential in selecting useful information for CCG supertagging.

ID	Settings			Tag	LF
	In-chunk	Cross-chunk	Attention		
1	√	√	√	**96.39**	**90.68**
2	×	√	√	96.34	90.58
3	√	×	√	96.30	90.50
4	√	√	×	87.31	82.02
5	×	√	×	95.01	89.84
6	√	×	×	89.83	84.57
7	×	×	×	96.25	90.46

Table 5: Experimental results of models with uncased BERT-Large encoder on the test set of CCGbank, where the in-chunk, cross-chunk edges or the attention mechanism in our A-GCN module is ablated.

4 Conclusion

In this paper, we propose A-GCN for CCG supertagging, with its graph built from chunks extracted from a lexicon. We use two types of edges for the graph, namely, in-chunk and cross-chunk edges for word pairs within and across chunks, respectively, and propose an attention mechanism to distinguish the important word pairs according to their contribution to CCG supertagging. Experimental results and the ablation study on the English CCGbank demonstrate the effectiveness of our approach to CCG supertagging, where state-of-the-art performance is obtained on both CCG supertagging and parsing. Further analysis is performed to investigate using different types of edges, which reveals their quality and confirms the necessity of introducing attention to GCN for CCG supertagging.

References

Sebastian Beschke. 2019. Exploring graph-algebraic CCG combinators for syntactic-semantic AMR parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 112–121, Varna, Bulgaria.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. 2018. Semi-supervised Sequence Modeling with Cross-view Training. *arXiv preprint arXiv:1809.08370*.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question Answering by Reasoning Across Documents with Graph Convolutional Networks. 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 2306–2317, Minneapolis, Minnesota.

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.

Kilian Evang, Lasha Abzianidze, and Johan Bos. 2019. CCGweb: a New Annotation Tool and a First Quadrilingual CCG Treebank. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 37–42, Florence, Italy.

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.

Binxuan Huang and Kathleen Carley. 2019. Syntax-Aware Aspect Level Sentiment Classification with Graph Attention Networks. 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 5469–5477, Hong Kong, China.

Chang-Ning Huang and Yan Song. 2015. Chinese CCGbank Construction from Tsinghua Chinese Treebank. *Journal of Chinese Linguistics Monograph Series*, (25):274–311.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text Level Graph Neural Network for Text Classification. 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 3444–3450, Hong Kong, China.

Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. 2017. Chunk-based Decoder for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1901–1912, Vancouver, Canada.

Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. 2010. Faster Parsing by Supertagger Adaptation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 345–355.

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.

Mike Lewis and Mark Steedman. 2014a. 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.

Mike Lewis and Mark Steedman. 2014b. Improved CCG Parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark.

Yan Song, Chang-Ning Huang, and Chunyu Kit. 2012. Construction of Chinese CCGbank. *Journal of Chinese Information Processing*, (3):2.

Yan Song, Chunyu Kit, and Xiao Chen. 2009. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, Suntec, Singapore.

Yan Song and Fei Xia. 2012. Using a Goodness Measurement for Domain Adaptation: A Case Study on Chinese Word Segmentation. In *LREC*, pages 3853–3860.

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.

Yuanhe Tian, Yan Song, Xiang Ao, Fei Xia, Xiaojun Quan, Tong Zhang, and Yonggang Wang. 2020a. Joint Chinese Word Segmentation and Part-of-speech Tagging via Two-way Attentions of Auto-analyzed Knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8286–8296, Online.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020b. Improving Constituency Parsing with Span Attention. In *Findings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. 2020c. Improving Chinese Word Segmentation with Wordhood Memory Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285.

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.

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.

Seunghyun Yoon, Joongbo Shin, and Kyomin Jung. 2018. Learning to Rank Question-Answer Pairs Using Hierarchical Recurrent Encoder with Latent Topic Clustering. 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 1575–1584, New Orleans, Louisiana.

Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. 2018. Consistent CCG parsing over multiple sentences for improved logical reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 407–412, New Orleans, Louisiana.

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.

Hongming Zhang, Yan Song, and Yangqiu Song. 2019. Incorporating Context and External Knowledge for Pronoun Coreference Resolution. 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 872–881, Minneapolis, Minnesota.

Appendix A: Example Sentences with Extracted Chunks

(1) No , it was n't Black Monday .

(2) The 49 stock specialist firms on the Big Board floor -- the buyers and sellers of last resort who were criticized after the 1987 crash -- once again could n't handle the selling pressure .

(3) Arbitragers could n't dump their UAL stock -- but they rid themselves of nearly every rumor stock they had .

(4) Traders said most of their major institutional investors , on the other hand , sat tight .

(5) At 3:30 p.m. , at the end of the cooling off period , the average was down 114.76 points .

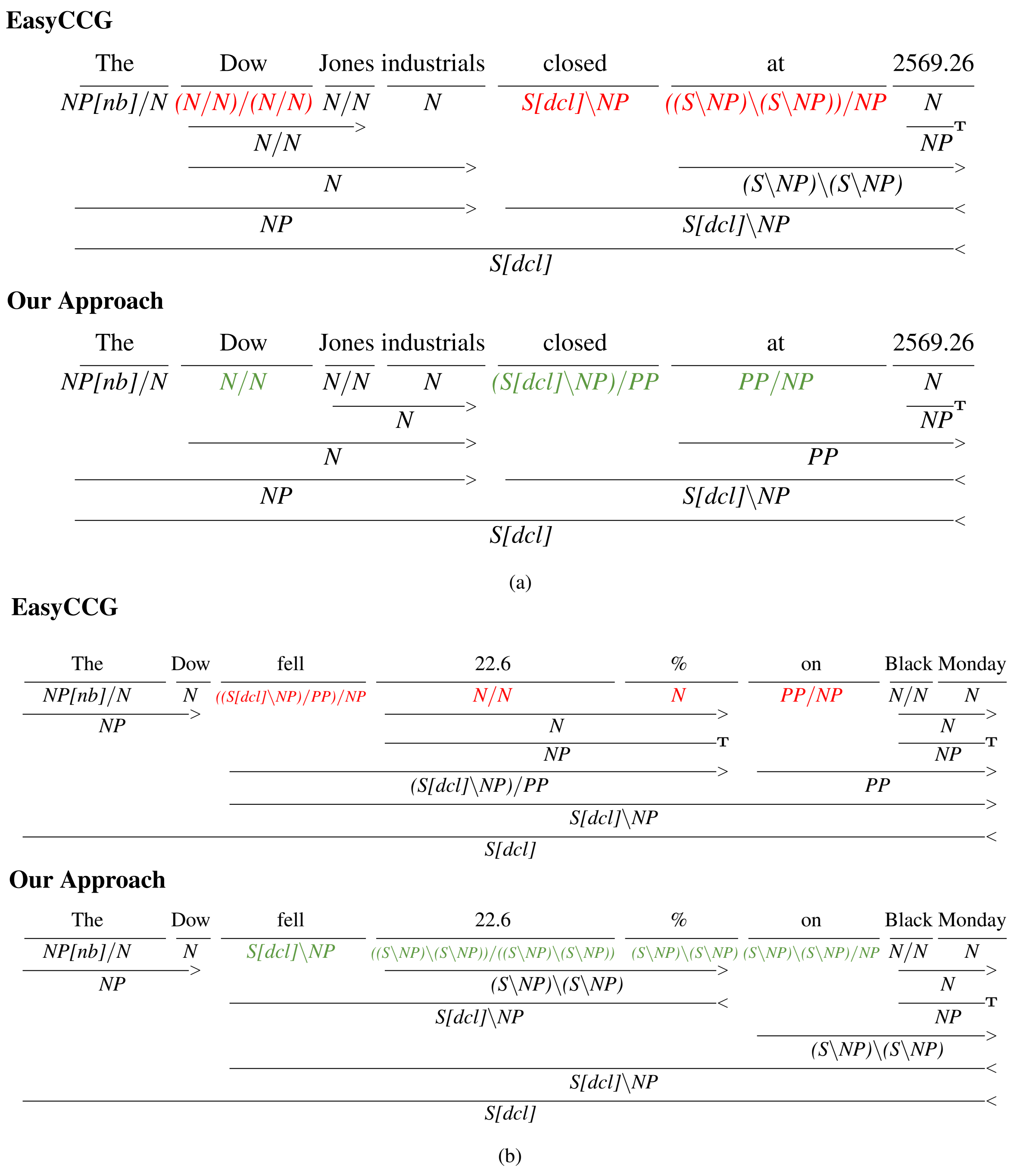
Figure 3: Example sentences with the chunks extracted from the lexicon $\mathcal{N}$ highlighted in green.

In the main experiments, we use the lexicon obtained from the training set of the English CCGbank to extract chunks in each sentence, where the chunks are used to build the graph. Figure 3 shows five example sentences in which the extracted chunks are highlighted in green. We report more examples in the supplemental materials.

Appendix B: Example Suppertagging and Parsing Results

Figure 4 shows the CCG supertagging and parsing results of EasyCCG[8] and our approach (i.e., BERT + A-GCN (Chunk)) on two example sentences. In the figure, the correct and incorrect supertags are represented by green and red color, respectively. We report more CCG parsing results of our approach in the supplemental materials.

[8]We use the implementation from CCGweb (Evang et al., 2019) at `https://ccgweb.phil.hhu.de/`.


EasyCCG
The Dow Jones industrials closed at 2569.26
Our Approach
The Dow Jones industrials closed at 2569.26
(a)
EasyCCG
The Dow fell 22.6 % on Black Monday
Our Approach
The Dow fell 22.6 % on Black Monday
(b)
Figure 4: The CCG supertagging and parsing results of EasyCCG and our approach (i.e., BERT + A-GCN (Chunk)) on the examples, where the correct and incorrect supertags are represented in green and red color, respectively.