Lexicon-Enhanced Dual-Channel Graph Attention Networks for End-to-End Chinese Parsing

Zhiyang Teng$^{\alpha,\gamma}$,Yuan Zhang$^{\beta}$, Yue Zhang$^{\alpha,\gamma}$

α School of Engineering, Westlake University
β Microsoft
γ Institute of Advanced Technology, Westlake Institute for Advanced Study

Abstract

Chinese parsing has traditionally been solved by three pipeline modules including word segmentation, part-of-speech tagging and dependency parsing. In this paper, we propose an end-to-end Chinese parsing model based on *character inputs* which jointly learns to output word segmentation, part-of-speech tags and dependency structures. In particular, our parsing model relies on dual-channel graph attention networks (DC-GAT), which can enrich the character inputs with structural and semantic channels from *external lexicon knowledge*. Experiments on three Chinese parsing benchmarks show the effectiveness of our models, achieving the state-of-the-art results on end-to-end Chinese parsing.

Keywords: Graph Attention Networks, End-to-End Chinese Parsing, Chinese Word Segmentation, Chinese Part-of-Speech Tagging, Chinese Dependency Parsing
2010 MSC: 00-01, 99-00

1. Introduction

As a fundamental task in syntactic analysis, dependency parsing has received much research attention (Dozat and Manning, 2017; Strubell et al., 2018; Ma et al., 2018; Li et al., 2019). It offers useful information to a range of downstream tasks, such as

⋆Yuan Zhang and Zhiyang Teng equally contribute to this work. This work was done while Yuan Zhang was a research assistant in Westlake University.

Email address: `tengzhiyang@westlake.edu.cn, chuengyuenn@gmail.com, yue.zhang@wias.org.cn` (Zhiyang Teng$^{\alpha,\gamma}$,Yuan Zhang$^{\beta}$, Yue Zhang$^{\alpha,\gamma}$)

[1]Yue Zhang is the corresponding author.

relation extraction (Gamallo et al., 2012; Miwa and Bansal, 2016; Guo et al., 2019; Zhang et al., 2018), semantic parsing (Poon and Domingos, 2009; Sun et al., 2018) and machine translation (Bohnet, 2010). As shown in Figure 1, the goal of syntactic dependency parsing is to build a dependency tree for a given sentence, where each arc represents a head-dependent relationship between two words, such as the "subject" relation between the words "副总统(vice president)" and "表示(express)".

Chinese sentence are written as sequences of characters. Traditionally, Chinese dependency parsing takes word segmentation and part-of-speech (POS) tagging as pre-processing steps (Zhou, 2000; Ma and Zhao, 2012; Zhang and McDonald, 2014). For example, given the sentence "伊朗副总统对访华成果表示满意(Iranian vice president was satisfied with the results of the visit to China)", the first step is to segment the sequence into "伊朗(Iranian) 副总统(vice president) 对(for) 访(visit) 华(China) 成果(result) 表示(express) 满意(satisfy) " and assign POS tag for each word, e.g. "Noun(NN)" for the word "成果(result)". The pipeline method, however, suffers from error propagation as incorrect word boundaries and POS tags lead to decreases in parsing performance. To address the problem, end-to-end models have been investigated, which take character sequences as input and jointly perform word segmentation, POS tagging and dependency parsing via multi-task learning (Hatori et al., 2012; Zhang et al., 2013; Kurita et al., 2017; Li et al., 2018).

We consider a graph-based method for end-to-end parsing, adopting the bi-affine framework of Dozat and Manning (2017), which first encodes the input sentences with neural encoders and then applies biaffine scoring functions to calculate the arc scores for the dependency relation classifiers. One salient difference of neural graph-based parsing, as compared with its transition-based counterpart (Chen and Manning, 2014; Andor et al., 2016), is that the representation of input is calculated first, before predicting local outputs such as sequence labels and bi-word relation tags. However, for a joint graph-based parser, word segmentation and dependency parsing are performed jointly on characters, and as a consequence word information cannot be directly used to benefit parser disambiguation. To solve this issue, we consider integrating lexicon knowledge for enriching the character sequence representation, by encoding a lattice structure, which considers both the characters and all the words in the input that match

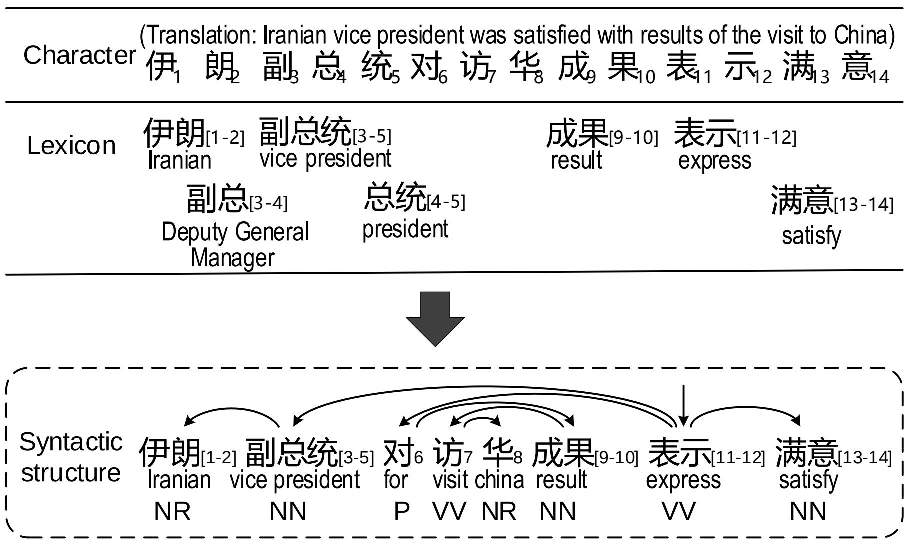

Figure 1: End-to-end parsing exploiting lexicons.

a dictionary. Our method is inspired by recent work on Chinese NER (Gui et al., 2019; Li et al., 2020; Tang et al., 2020), which shows that encoding potential words together with input characters can allow more informative model decisions. For example, in Figure 1, the lexicon entry “副总统(vice president)” is a correct word for the input sentence while “副总(deputy general manager)” is not. The model is designed to support automatic selection of the correct word “副总统(vice president)” with more attentions to resolve the semantic ambiguities.

In particular, we investigate both bidirectional long short-term memory networks (BiLSTMs) and self attention networks (SANs), by making use of lattice LSTMs (Zhang et al., 2019), and building a graph attention networks (GAT; Veličković et al. (2017)), respectively. The latter runs over a order of magnitude faster than the former thanks to strong parallelization. We propose a dual-channel GAT (DCGAT) to integrate information from lexicon words into this character-based GAT (i.e., Transformer (Vaswani et al., 2017)) by taking them as additional vertices in the graph, adding word-character edges and word-word edges to the input graph. The standard self-attention function is further extended into a novel combination of a **structural channel** and a

semantic channel, the former taking paths in the graph into considerations and the latter using semantic similarities for weight calculations.

Experiments on three Chinese parsing datasets show that integrating lexicon word information by DCGAT is useful for improving character-level end-to-end parsing. Our graph-based parser, enriched with word-level features, outperforms all existing methods in all the datasets, achieving the best results on segmentation, POS tagging and parsing in the literature. To our knowledge, we are the first to investigate end-to-end Chinese parsing exploiting lexicon knowledge. We will release our code and models at https://github.com/zeeeyang/chn_e2e_parser.

2. Related Work

End-to-End Chinese Parsing. Hatori et al. (2012) pioneer research on the joint model of word segmentation, POS tagging, and dependency parsing for Chinese using transition-based methods. Zhang et al. (2014) exploit the manually annotated intra-character dependencies. Zhang et al. (2015) consider joint word segmentation, POS tagging and dependency reranking using randomized greedy inference. Kurita et al. (2017) first investigate joint Chinese lexical and syntactic analysis with neural models. All the above methods are transition-based. In contract, we investigate the same task, but our end-to-end models are built on graph-based parsers.

Recently, Yan et al. (2020) consider using a BiLSTM and BERT encoder for representing character sequences for end-to-end Chinese parsing. Our work is similar to them in being a graph-based parser, but differs in three aspects. First, we investigate the effectiveness of word information for the task by considering a novel graph attention network with semantic and structural channels. Second, we compare BiLSTM encoding and Transformer encoding with BERT. Third, while they consider joint segmentation and parsing, we consider joint segmentation, POS-tagging and parsing. Wu and Zhang (2021) improves the Transformer encoder with relative position embedding for graph-based end-to-end Chinese character-level parsing. However, they do not consider external lexicon knowledge. In addition, their model designs an adapter Houlsby et al. (2019) to improve BERT, while our model is directly based on original BERT.

They also leverage dependency labels as supervision signals during training, which is slightly different from previous baseline Zhang et al. (2014); Yan et al. (2020).

Word-character Lattice Neural Networks. Chen et al. (2017) and Zhang and Yang (2018) use lattice LSTMs to deal with mixed word and character inputs. Ding et al. (2019) use graph convolutional networks for entity lattice inputs. Their lattice requires named entities and their entity types as inputs. Gui et al. (2019) designs a global recurrent network to integrate lexicon features for Chinese NER. Tang et al. (2020) proposes a word-char graph convolutional network with dense connections between graph convolutional layers for Chinese NER. More closely related to our work, Sperber et al. (2019) adapt Transformer (Vaswani et al., 2017) for lattice inputs. Our models are different from them in two aspects. First, their lattice is built from compressing speech hypothesis, while we build the lattice by lexicon matching. Second, there is no concept of word in their models, while we explicitly model word inputs and exploit pretrained word embeddings. Li et al. (2020) proposes a FLat-Lattice Transformer for Chinese NER. They mainly use a unified position embeddings to feed the relations between words and characters to Transformer structures. Different from them, we explicitly differentiate semantic and structural information according to the edge types in lattice by taking inspirations from graph Transformers (Veličković et al., 2017). The above methods can be regarded as a variant of our models, which only captures semantic features. To our knowledge, we are the first to use lattice neural networks for parsing.

3. Task Description and Overall Framework

Formally, as shown in Figure 1, given a sequence of characters $s = c_1, ..., c_n$, the goal of end-to-end parsing is to obtain a dependency tree $T = (V, E)$ together with a segmented word sequence $s_w = w_1, ..., w_k$ of s and a corresponding POS tag sequence $s_{pos} = pos_{w_1}, ..., pos_{w_k}$, where pos_{w_1} is the POS tag of the i-th word. The node set V contains all segmented words in s_w and a root dummy node. The arc set $E = \{(w_i, head_{w_i})\}$, where $head_{w_i}$ represents the head word of the i-th word w_i in a dependency relation. Figure 2 shows the overall framework of our method. In general, we adopt graph-based parsing framework of Dozat and Manning (2017). Given an

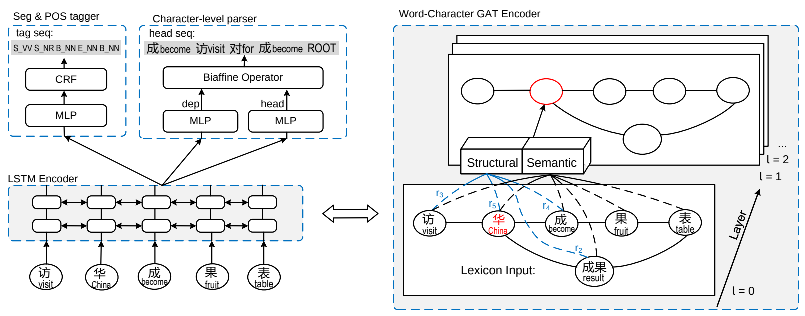

Figure 2: End-to-end Chinese parsing exploiting lexicons. The input is a part of the sentence in Figure 1 with character index $i \in [7, 11]$. The output sequences are the gold labels accordingly. Figure 1 shows the output structure for the entire sentence.

input character sequence s, we construct a lattice G_L by matching the subsequence of s from a external word dictionary D to enrich the input. Based on s and G_L, we adopt neural encoders to obtain representation vectors for each input character. Using the character representation vectors, we leverage biaffine transformation for dependency arc prediction. For word segmentation and POS tagging, a CRF layer is used to model the label dependencies of the input sequence. For the encoder, we investigate both LSTM-based (Section 4) and self-attention based (Section 5) graph neural networks to exploit the external lexicon knowledge, both of which have been considered for dependency parsing (Kiperwasser and Goldberg, 2016; Li et al., 2019), and for mixing char and word information for Chinese NER Zhang and Yang (2018); Li et al. (2020) .

4. LSTM-based Encoders

LSTM. We first consider LSTM-based encoders for learning input representations without using external word dictionaries, following Dozat and Manning (2017), Kiperwasser and Goldberg (2016) and Wang and Chang (2016).

Specifically, we obtain contextual character representation vectors through BERT (Devlin et al., 2019):

$$\mathbf{e}_{c_1}, \mathbf{e}_{c_2}, \ldots, \mathbf{e}_{c_n} = \text{BERT\_Encoder}(c_1, c_2, \ldots, c_n). \tag{1}$$

Then we use a multi-layer bi-directional LSTM structure to calculate the character sequence representations. In particular, the initial character embedding $\mathbf{e}_{c_i}$ are denoted as $\mathbf{h}^0_{c_i}$. Subsequently, for the k-th layer, $\mathbf{h}^k_{c_1}, ..., \mathbf{h}^k_{c_n}$ are calculated from $\mathbf{h}^{k-1}_{c_1}, ..., \mathbf{h}^{k-1}_{c_n}$ as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{h}^k_i}, \overrightarrow{\mathbf{c}^k_i} &= \overrightarrow{\textit{LSTM}^k}(\overrightarrow{\mathbf{h}^{k-1}_i}, \overrightarrow{\mathbf{h}^k_{i-1}}, \overrightarrow{\mathbf{c}^k_{i-1}}), \\
\overleftarrow{\mathbf{h}^k_i}, \overleftarrow{\mathbf{c}^k_i} &= \overleftarrow{\textit{LSTM}^k}(\overleftarrow{\mathbf{h}^{k-1}_i}, \overleftarrow{\mathbf{h}^k_{i+1}}, \overleftarrow{\mathbf{c}^k_{i+1}}), \\
\mathbf{h}^k_i &= \langle \overleftarrow{\mathbf{h}^k_i}, \overrightarrow{\mathbf{h}^k_i} \rangle, \quad \mathbf{c}^k_i = \langle \overleftarrow{\mathbf{c}^k_i}, \overrightarrow{\mathbf{c}^k_i} \rangle,
\end{aligned}$$

where $\textit{LSTM}^k$ is the LSTM network for the k-th layer, $\mathbf{h}^k_i$ and $\mathbf{c}^k_i$ are the hidden and cell states of the i-th character respectively. $\rightarrow$ and $\leftarrow$ indicate the left and right composition directions of LSTM networks respectively. The final representation of the i-th character c_i is the output hidden vector of the K-th layer: $\mathbf{h}_{c_i} = \mathbf{h}^K_{c_i}$.

To be more specific, the LSTM cell $\overrightarrow{\textit{LSTM}^k}(\overrightarrow{\mathbf{h}^{k-1}_i}, \overrightarrow{\mathbf{h}^k_{i-1}}, \overrightarrow{\mathbf{c}^k_{i-1}})$ is given by,

$$\begin{aligned}
\begin{bmatrix} \overrightarrow{\mathbf{i}^k_i} \\ \overrightarrow{\mathbf{o}^k_i} \\ \overrightarrow{\mathbf{f}^k_i} \\ \overrightarrow{\widetilde{\mathbf{c}}^k_i} \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\overrightarrow{\mathbf{W}^{k}}^{\top} \begin{bmatrix} \overrightarrow{\mathbf{h}^k_{i-1}} \\ \overrightarrow{\mathbf{h}^{k-1}_i} \end{bmatrix} + \overrightarrow{\mathbf{b}^k} \right) \\
\overrightarrow{\mathbf{c}^k_i} &= \overrightarrow{\mathbf{f}^k_i} \odot \overrightarrow{\mathbf{c}^k_{i-1}} + \overrightarrow{\mathbf{i}^k_i} \odot \overrightarrow{\widetilde{\mathbf{c}}^k_i} \\
\overrightarrow{\mathbf{h}^k_i} &= \overrightarrow{\mathbf{o}^k_i} \odot \tanh(\overrightarrow{\mathbf{c}^k_i})
\end{aligned} \tag{2}$$

where $\overrightarrow{\mathbf{i}^k_i}$, $\overrightarrow{\mathbf{o}^k_i}$ and $\overrightarrow{\mathbf{f}^k_i}$ are input, output and forget gates, respectively. $\overrightarrow{\mathbf{W}^{k}}^{\top}$ and $\overrightarrow{\mathbf{b}^k}$ are model parameters. σ denotes the sigmoid function. Similarly, $\overleftarrow{\textit{LSTM}^k}(\overrightarrow{\mathbf{h}^{k-1}_i}, \overrightarrow{\mathbf{h}^k_{i-1}}, \overrightarrow{\mathbf{c}^k_{i-1}})$ can be defined in the same way as $\overrightarrow{\textit{LSTM}^k}(\overrightarrow{\mathbf{h}^{k-1}_i}, \overrightarrow{\mathbf{h}^k_{i-1}}, \overrightarrow{\mathbf{c}^k_{i-1}})$ by using parameters $\overleftarrow{\mathbf{W}^{k}}^{\top}$ and $\overleftarrow{\mathbf{b}^k}$.

Lattice LSTM. We follow lattice LSTM structure (Zhang and Yang, 2018) to integrate word features from a external dictionary. Formally, the hidden states are calculated as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{h}^k_i}, \overrightarrow{\mathbf{c}^k_i} &= \overrightarrow{\textit{LatticeLSTM}^k}(\overrightarrow{\mathbf{h}^{k-1}_i}, \overrightarrow{\mathbf{h}^k_{i-1}}, \overrightarrow{\mathbf{c}^k_{i-1}}, \mathbf{E}_{w_{\cdot i}}), \\
\overleftarrow{\mathbf{h}^k_i}, \overleftarrow{\mathbf{c}^k_i} &= \overleftarrow{\textit{LatticeLSTM}^k}(\overleftarrow{\mathbf{h}^{k-1}_i}, \overleftarrow{\mathbf{h}^k_{i+1}}, \overleftarrow{\mathbf{c}^k_{i+1}}, \mathbf{E}_{w_{i\cdot}}), \\
\mathbf{h}^k_i &= \langle \overleftarrow{\mathbf{h}^k_i}, \overrightarrow{\mathbf{h}^k_i} \rangle, \quad \mathbf{c}^k_i = \langle \overleftarrow{\mathbf{c}^k_i}, \overrightarrow{\mathbf{c}^k_i} \rangle,
\end{aligned}$$

where *LatticeLSTM*^{k} is the Lattice-LSTM for the k-th layer. $\mathbf{E}_{w_{\cdot i}}$ and $\mathbf{E}_{w_{i\cdot}}$ are the sets of word embeddings for words ending and starting with the i-th character, respectively. Specifically, we average the character-level representations to obtain the word-level representations. Formally, the representation vector of a matched lexicon entry $d_{b,e}$ is

$$\mathbf{d}_{b,e} = \frac{\sum_{i=b}^{e} \mathbf{e}_{c_i}}{e-b+1}. \tag{3}$$

Denote $\overrightarrow{\mathbf{h}_{b,e}^{k}}$ and $\overrightarrow{\mathbf{c}_{b,e}^{k}}$ as the word-level hidden state and cell state vectors respectively to represent the recurrent state of $\mathbf{d}_{b,e}$ from the start of the sentence at the k-th layer. $\overrightarrow{\mathbf{h}_{b,e}^{0}} = \mathbf{d}_{b,e}$ The value of $\overrightarrow{\mathbf{c}_{b,e}^{k}}$ is calculated by:

$$\begin{aligned}\begin{bmatrix} \overrightarrow{\mathbf{i}_{b,e}^{k}} \\ \overrightarrow{\mathbf{f}_{b,e}^{k}} \\ \overrightarrow{\widetilde{c}_{b,e}^{k}} \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\overrightarrow{\mathbf{W}_{w}^{k}}^{\top} \begin{bmatrix} \overrightarrow{\mathbf{h}_{b,e}^{k-1}} \\ \overrightarrow{\mathbf{h}_{b}^{k}} \end{bmatrix} + \overrightarrow{\mathbf{b}_{w}^{k}} \right) \\ \overrightarrow{\mathbf{c}_{b,e}^{k}} &= \overrightarrow{\mathbf{f}_{b,e}^{k}} \odot \overrightarrow{\mathbf{c}_{b}^{k}} + \overrightarrow{\mathbf{i}_{b,e}^{k}} \odot \overrightarrow{\widetilde{c}_{b,e}^{k}}\end{aligned} \tag{4}$$

where $\overrightarrow{\mathbf{i}_{b,e}^{k}}$ and $\overrightarrow{\mathbf{f}_{b,e}^{k}}$ are input and forget gates, respectively. $\overrightarrow{\mathbf{W}_{w}^{k}}$ and $\overrightarrow{\mathbf{b}_{w}^{k}}$ are model parameters. In this way, the character-level recurrent information $\overrightarrow{\mathbf{h}_{b}^{k}}$ of the starting character indexed by b of the matched word $w_{b,e}$ is propagated into the word-level cell $\overrightarrow{\mathbf{c}_{b,e}^{k}}$.

Similarly, the information flow can also be forwarded from the word-level cell $\overrightarrow{\mathbf{c}_{b,e}^{k}}$ to the character-level cell $\overrightarrow{\mathbf{c}_{e}^{k}}$ of the ending character indexed by e of the matched word $w_{b,e}$. Since there can be multiple recurrent paths for such kind of information flow for each $\overrightarrow{\mathbf{c}_{e}^{k}}$, lattice LSTMs introduce input gates $\overrightarrow{\mathbf{i}_{b,e}^{k}}$ to control the information flow,

$$\overrightarrow{\mathbf{i}_{b,e}^{k}} = \sigma\left(\overrightarrow{\mathbf{W}}^{l\top} \begin{bmatrix} \overrightarrow{\mathbf{h}_{e}^{k-1}} \\ \overrightarrow{\mathbf{c}_{b,e}^{k}} \end{bmatrix} + \overrightarrow{\mathbf{b}_{l}^{k}},\right) \tag{5}$$

where $\overrightarrow{\mathbf{W}^{l}}$ and $\overrightarrow{\mathbf{b}_{l}^{k}}$ are model parameters. The character-level cell values $\overrightarrow{\mathbf{c}_{e}^{k}}$ is then given by

$$\begin{aligned}
\overrightarrow{\boldsymbol{\alpha}_{b,e}^{k}} &= \frac{\exp(\overrightarrow{\mathbf{i}_{b,e}^{k}})}{\exp(\overrightarrow{\mathbf{i}_{e}^{k}}) + \sum_{b' \in \{b'' | w_{b'',j} \in \mathbb{D}\}} \exp(\overrightarrow{\mathbf{i}_{b',e}^{k}})}, \\
\overrightarrow{\boldsymbol{\alpha}_{e}^{k}} &= \frac{\exp(\overrightarrow{\mathbf{i}_{k}^{e}})}{\exp(\overrightarrow{\mathbf{i}_{e}^{k}}) + \sum_{b' \in \{b'' | w_{b'',j} \in \mathbb{D}\}} \exp(\overrightarrow{\mathbf{i}_{b',e}^{k}})}, \\
\overrightarrow{\mathbf{c}_{e}^{k}} &= \sum_{b \in \{b' | w_{b',j} \in \mathbb{D}\}} \overrightarrow{\boldsymbol{\alpha}_{b,e}^{k}} \odot \overrightarrow{\boldsymbol{c}_{b,e}^{k}} + \overrightarrow{\boldsymbol{\alpha}_{e}^{k}} \odot \overrightarrow{\widetilde{\boldsymbol{c}}_{e}^{k}}
\end{aligned} \tag{6}$$

$\overrightarrow{\boldsymbol{\alpha}_{b,e}^{k}}$ and $\overrightarrow{\boldsymbol{\alpha}_{e}^{k}}$ are normalized input gates. The final hidden vectors $\overrightarrow{\mathbf{h}_{e}^{k}}$ are still computed as described by Eq 2. Similarly, $\overleftarrow{\mathit{LatticeLSTM}^{k}}(\overleftarrow{\mathbf{h}_{i}^{k-1}}, \overleftarrow{\mathbf{h}_{i+1}^{k}}, \overleftarrow{\mathbf{c}_{i+1}^{k}}, \mathbf{E}_{w_{i\cdot}})$ can be defined and $\overleftarrow{\mathbf{h}_{e}^{k}}$ can be derived.

5. Dual-Channel GAT Encoder

For the convenience of describing both word and characters, we unify both w and c as nodes in a graph represented using a graph attention network (GAT; (Veličković et al., 2017)), integrating lattice structures into Transformer structures. We propose a dual-channel GAT (DCGAT) by defining two channels for capturing semantic and structural features in the graph, respectively. In particular, the **semantic channel** captures interaction between characters and words in the sentence without differentiating their types, and the **structural channel** adds the type and relative position when considering node interactions. The detailed definitions will be given later. As shown in Figure 2, DCGAT encoder is used as an alternative encoder for the LSTM encoder component in the earlier section to model a lattice structure. The model consists of a multi-layer encoder. Each layer consists of two sub-layers, including a multi-head self-attention sublayer and a position-wise feed-forward network sublayer. Layer normalization and residual network are used for each sublayer. DCGAT works by iterative updating the representation of each node through layers. In each layer, each node receives information from its neighbors in the input graph structure to update its representation vector.

In general, for a GAT, the representation vector of the i-th node t_i for the m-th head is updated by

$$\mathbf{h}_i^m = \sum_{j \in \text{neigbors(i)}} \frac{\exp(s_{ji}^m)}{\sum_{j'} \exp(s_{j'i}^m)} \cdot \left(\mathbf{x}_j \mathbf{W}^{V,m}\right),$$

where $\mathbf{x}_j$ is the input representation vector of t_j for the current encoder layer, and $\mathbf{W}^{V,m}$ is a model parameter. The final output $\mathbf{h}_i$ for parsing is the concatenation of the outputs from all the M attention heads of the last layer as given by the above equation: $\mathbf{h}_i = [\mathbf{h}_i^1, ..., \mathbf{h}_i^M]$.

s_{ji}^m is a similarity score between node j and node i. In particular, we decompose the similarity score into semantic channels and structural channels:

$$s_{ji}^m = s_{ji}^{sem,m} + s_{ji}^{str,m},$$

where $s_{ji}^{sem,m}$ and $s_{ji}^{str,m}$ are the similarity scores obtained from semantic channels and structural channels, respectively.

Semantic Channel. Formally, denote the input character-word mixed sequence of the input lattice as $t = [c_1, ..., c_n, w_1, ..., w_k]$, where $c_1, ..., c_n$ represent the input character sequence and $w_1, ..., w_k$ represent the words in the input that match a lexicon. Denote the input vectors of each layer as $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_n, \mathbf{x}_{n+1}, ..., \mathbf{x}_{n+k}]$, where $\mathbf{x}_i$ is the input representation vector of c_i when $1 \leq i \leq n$ or the input representation of w_i when $n + 1 \leq i \leq n + k$.

The input for the first encoder layer is the embedding sequence $[\mathbf{e}'_{c_1}, ..., \mathbf{e}'_{c_n}, \mathbf{e}'_{w_1}, ..., \mathbf{e}'_{w_k}]$. We use static position embeddings to encode the input (Vaswani et al., 2017). $e^p(i)$ represents the position embedding for the i-th position. We inject position embeddings to characters and word input as follows:

$$\mathbf{e}'_{c_i} = \mathbf{e}_{c_i} + \mathbf{e}^p(i); \mathbf{e}'_{w_i} = \mathbf{e}_{w_i} + \mathbf{e}^p(b_{w_i}),$$

where $\mathbf{e}_{c_i}$ is the contextual representation vector of the i-th character from BERT and we average the representation vectors of characters inside w_i to obtain $\mathbf{e}_{w_i}$ as the word representation vector for the i-th word. We take the position of the beginning character b_{w_i} of the word w_i in the original sentence to represent the position of w_i.

For the m-th attention head, the similarity between two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ can be calculated by a vector inner product:

$$s_{ji}^{sem,m} = \left(\mathbf{x}_i \mathbf{W}^{K,m}\right) \cdot \left(\mathbf{x}_j \mathbf{W}^{Q,m}\right),$$

where $\mathbf{W}^{K,m}$ and $\mathbf{W}^{Q,m}$ are the model parameters for the m-th attention head.

The similarity $s_{ji}^{sem,m}$ can control how much information the i-th node can receive from the j-th node. The key in the above information exchange process is a mixed self-attention mechanism between character and words, where a weight score s_{ji}^{sem} is calculated by the similarity of nodes t_j and t_i, which can both be characters or words. To this end, s_{ji}^{sem} can be regarded as a **semantic channel**.

Structural Channel. We further introduce a **structural channel** by taking path structure into consideration. As shown in Table 1, we differentiate edges according to both the word/character difference and the relative position, resulting in 7 types of edges. For example, "word $\rightarrow$ character" represents an edge from a word to a character at its right. "self-to-self" represents a self-loop over a character or a word.

We make use of rich edge types by defining structural channels. Each token t_i can receive information from all the input according to the edges defined in Table 1. Taking the character "华(China)" in Figure 2 as an example, it can receive information from "成果(result)", "访(visit)", "成(become)", "华(China)" through the edges r_2, r_3, r_4 and r_5 in the table, respectively. In addition, we define a special edge "others" for each other relation type so that "华(China)" can receive information from all characters and words.

Formally, denote the relation from node j to node i as r_{ji}. We can obtain the embedding of each relation r_{ji} through an embedding lookup table: $\mathbf{e}_{ji}S^r = \text{emb}^r(r_{ji})$. We further calculate contextualized relation embeddings $\tilde{\mathbf{e}}_{ji}$, which is sensitive to the representations of the two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ for r_{ji}:

$$\tilde{\mathbf{e}}_{ji}^{m} = \sigma(\mathbf{e}_{ji}\mathbf{W}^{E,m} + \mathbf{e}_{x_i}\mathbf{W}^{S,m} + \mathbf{e}_{x_j}\mathbf{W}^{T,m}),$$

where $\mathbf{W}^{E,m}$, $\mathbf{W}^{S,m}$ and $\mathbf{W}^{T,m}$ are model parameters, and σ is a sigmoid function.

Block		Relation	Example
A (char)	1	word $\rightarrow$ char	-
	2	char $\leftarrow$ word	成果(result):华(China)
	3	char $\rightarrow$ char	访(visit):华(China)
	4	char $\leftarrow$ char	成(become):华(China)
	5	self-to-self	华(China):华(China)
B (word)	6	char $\rightarrow$ word	华(China):成果(result)
	7	word $\leftarrow$ char	表(table):成果(result)
	8	self-to-self	成果(result):成果(result)

Table 1: Types of relations of word-character interactions. Relation expression "A $\rightarrow$ B" indicates a relation from A (left) to B (right). Block A (to a character) and Block B (to a word) take the character "华(China)" and word "成果(Result)" as examples, respectively.

The structural similarity score is then calculated as:

$$s_{ji}^{str,m} = \mathbf{w}^m \cdot \tilde{\mathbf{e}}_{ji}^m,$$

where vector $\mathbf{w}^m$ is a parameter for the m-th attention head.

6. Decoding and Training

Output Layers. The output layers are the same for both the LSTM-based encoder and the DCGAT encoder. For word segmentation and POS tagging, we use a joint tag scheme $t_{ws}\_t_{pos}$, where $t_{ws} \in \{B, M, E, S\}$ denotes segmentation labels for *beginning*, *middle*, *ending* and *standalone* character of words respectively (Xue and Shen, 2003), and t_{pos} denotes POS tag. The probability P_{segpos} for the joint tag sequence *segpos* is calculated by a CRF layer:

$$\mathbf{t}_i = \mathrm{MLP}_t(\mathbf{h}_{c_i}),$$

$$P_{segpos} = \frac{\exp\left(\sum_i (\mathbf{W}_{CRF}^{segpos_i}\mathbf{t}_i + b_{CRF}^{(segpos_{i-1}, segpos_i)})\right)}{\sum_{y'} \exp\left(\sum_i (\mathbf{W}_{CRF}^{segpos'_i}\mathbf{t}_i + b_{CRF}^{(segpos'_{i-1}, segpos'_i)})\right)},$$

where $b_{CRF}^{(segpos'_{i-1}, segpos'_i)}$ is the adjacent tag transition parameter and $\mathbf{W}_{CRF}^{segpos_i}$ is the model parameter.

For dependency parsing, following Dozat and Manning (2017), we distinguish the head and dependent representations of each character. For the i-th character c_i, the head representation $\mathbf{h}_{c_i}$ and the dependent representation $\mathbf{d}_{c_i}$ are obtained through multi-layered perceptrons:

$$\mathbf{h}_i = \text{MLP}_h(\mathbf{h}_{c_i});\ \mathbf{d}_i = \text{MLP}_d(\mathbf{h}_{c_i})$$

The dependency confidence score dep_{ij} of the dependency relation $j \rightarrow i$ is obtained by using a biaffine transformation:

$$dep_{ij} = \text{softmax}_i \left(\mathbf{h}_j^T \mathbf{A} \mathbf{d}_i + \mathbf{b}_1^T \mathbf{h}_j + \mathbf{b}_2^T \mathbf{d}_i \right),$$

where $\mathbf{A}$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are model parameters.

Training. Given a training instance T, We compute a negative log-likelihood loss value $\mathcal{L}$ on each character c_i over the head probability dep_{ij} locally and accumulated along the sentence to train the model, plus the loss of joint segmentation and POS tags:

$$\mathcal{L} = -(\sum_{i=1}^{N} \log dep_{i,head_i} + \log P_{segpos}),$$

where $head_i$ and tag_i denote the head index and the joint segmentation-POS tag of the i-th character, respectively. During training, we minimize $\mathcal{L}$ to train the model parameters.

Decoding. We use hierarchical decoding for end-to-end dependency parsing. First, we perform a decoding to parse the internal structure of a word and find the root character, and then perform a decoding for all root characters in the sentence to obtain a word-level parsing tree. For example, suppose that the input character sentence is $c_1, c_2, c_3, c_4, c_5, c_6, c_7$. Figure 3 shows the decoding process. First, the word segmentor outputs three words $w_1 = [c_1, c_2, c_3], w_2 = [c_4, c_5], w_3 = [c_6, c_7]$. For each word w_i, the parser is used to obtain an inner parse. According to the inner parses, the root character for each word is decided. In the example, the root characters are c_1, c_4 and c_6 for w_1, w_2 and w_3, respectively. Then, we obtain word-level dependency structures

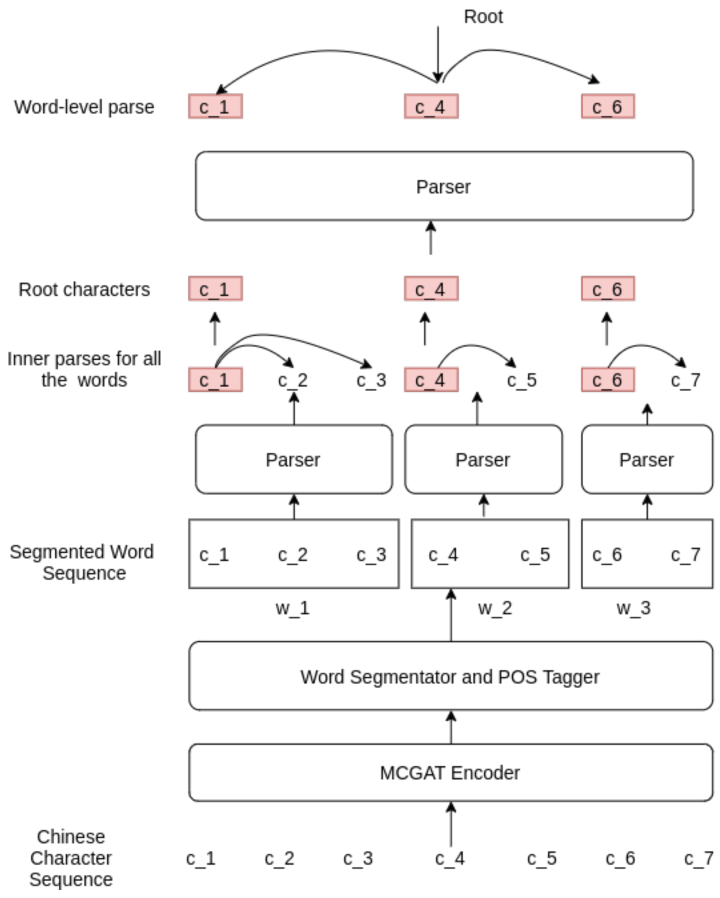

Figure 3: Hierarchical Decoding Process of End-to-End Chinese Parsing.

using root characters as shown in Figure 3. Based on the confidence score of all potential dependency arcs, the decoding process can be formulated as a max spanning tree (MST) problem which finds the highest scored set of arcs that form a dependency tree structure. Specifically, we use the Tarjan implementation of the Chu-Liu-Edmonds algorithm to find the MST derivation (Dozat and Manning, 2017).

7. Experiments

We investigate the effect of word information and our DCGAT encoder to graph-based end-to-end Chinese parsing.

Dataset	Split	#Sentence	#word
CTB5.0	train	18k	494k
	dev	350	6.8k
	test	348	8.0k
CTB6.0	train	23k	641k
	dev	2.1k	60k
	test	2.8k	82k
CTB7.0	train	31k	718k
	dev	10k	237k
	test	10k	245k

Table 2: Statistics of Datasets.

7.1. Settings

We use three releases of the Chinese Penn Treebank (i.e., 5.0, 6.0 and 7.0) (Xue et al., 2005), splitting the corpora into training, development and test sets according to previous work (Zhang et al., 2013). The dataset statistics are shown in Table 2. Following Hatori et al. (2012), the standard measures of word-level precision, recall and F1 scores are used to evaluate word segmentation, POS-tagging and dependency parsing, respectively. For a given word $w = c_1 c_2 ... c_k$, we simply use the right branching tree $c_1 \rightarrow c_2 \rightarrow ... \rightarrow c_k$ as the intra-word dependency structure.[2] Following Zhang and Yang (2018), we take the external Chinese lexicon dictionary from Song et al. (2018). The lexicon contains 167,405 words. 36.1% matched words on the CTB5 training set are gold words, and 35.9% matched words on the test set are gold words.

Hyper-parameters We search super-parameters manually on encoder layers, attention heads and dropout rate. The range of each super-parameters is: $[1, 3]$ for encoder layers, $[1, 16]$ for attention heads and $[0, 0.5]$ for dropout rate. The experiments are conducted under the same hardware environment: Geforce GTX 2080Ti graph card and

[2]In preliminary experiments, we also consider the left branching tree. We find that the simple right-branching tree gives the best results.

i7-7900 CPU. We initialize character embeddings using BERT (Devlin et al., 2019)[3], and word embeddings with the average pooling result of BERT character embeddings. Neither character nor word embeddings are fine-tuned due to limitation of GPU memory. The size of hidden states is set to 400 for all models (the hidden size of each direction in biLSTM is 200).

Training is done on mini-batches via Adagrad (Duchi et al., 2011) with a learning rate of 0.002, $\beta_1 = 0.9, \beta_2 = 0.9$ and $\epsilon = 1e^{-12}$. We adopt gradient clipping with a threshold of 5.0. Dropout (Hinton et al., 2012) is used on each layer, with a rate of 0.2.

7.2. Development Experiments

We conduct development experiments on the CTB 5.0 dev set to decide the final hyper-parameters of the two models. Below we show the details of two important factors.

Effect of Attention Heads Table 3 shows the effect of the number of attention head for the DCGAT model. We observe performance increases as the number of attention heads increases from 1 to 4, but adding more attention heads does not lead to further performance improvements. This coincides with the findings of analyzing Transformers (Voita et al., 2019; Michel et al., 2019) because only a few heads in multi-headed self attention module of Transformers play important roles. We thus set the number of attention heads to 4 accordingly.

Effect of Encoder Layers We investigate the effect of the number of encoder layers for DCGAT. The results are shown in Table 4. As the number of encoder layer increases from 1 to 3, performance improves for parsing, as well as word segmentation and POS tagging. We do not conduct experiments on more layers because of memory limitation. The number of encoder layers is set to 3 in the remaining experiments.

Effect of Word Information and Model Architectures We conduct a set of development experiments to verify the influence of model architecture and word information on each architecture. In particular, a character-level LSTM model gives 90.3% parsing accuracy scores. With word-character lattice LSTMs, the results are improved to

[3]https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

Setting	**SEG**	**POS**	**DEP**
1 head	98.5	96.4	91.0
4 heads	**98.8**	**96.5**	**91.4**
8 heads	98.6	96.4	91.3
16 heads	98.7	96.5	90.9
Character-level LSTM	98.4	96.1	90.3
Word-char lattice LSTM	98.7	96.4	90.8
Character-level Transformer	98.7	96.5	90.9
DCGAT	**98.8**	**96.5**	**91.4**

Table 3: Development experiments.

Parameter	**SEG**	**POS**	**DEP**
1 layer	98.5	96.2	90.0
2 layers	98.5	96.3	91.1
3 layers	98.8	96.5	91.4

Table 4: Development experiments for DCGAT.

90.8%. Using character-level Transformer, the model gives a 90.9% development accuracy for parsing, which is improved to 91.4% by using word information additionally in DCGAT. This shows that word information is beneficial to both the LSTM-based architecture and the self-attention-based architecture. In addition to dependency structures, the results for both word segmentation and POS-tagging are also improved. This shows that adding lexicon into a joint parsing model also benefits lexical subtasks.

In addition, LSTM-based models underperform their self-attention counterparts regardless whether word information is combined into the character encoder. We further compare the running time of the Lattice-LSTM model and DCGAT in both training and inference stages. Table 5 shows the runtime speeds on the CTB 5.0 dataset. We can find that the Lattice-LSTM model takes much more time than DCGAT. For example, for training, the Lattice-LSTM model takes more than half an hour for training one

Model	Train (s)	Test (s)
Lattice LSTM E2E	2,111	28
DCGAT	421	4

Table 5: Times of different models. The total training time for one epoch and the total evaluating time for test data for CTB 5.0 dataset are listed respectively.

epoch on CTB 5.0, and the total training takes several days, while the required time for DCGAT is 421s/epoch. This is mainly because the Lattice-LSTM model depends on sequential information paths for representation composition (Zhang and Yang, 2018), while DCGAT can deal with all inputs in parallel.

7.3. Final Results

7.3.1. Dependency Parsing Results

Table 6 shows the overall performances of end-to-end Chinese dependency parsing where our model is compared with the state-of-the-art methods in the literature. We report the performances on Chinese word segmentation, POS tagging and parsing, respectively. First, we can find that BERT pretrained models outperform neural models using traditional embeddings (Kurita et al., 2017; Li et al., 2018) and statistical models (Hatori et al., 2012; Zhang et al., 2014, 2015) by a large margin. Compared with Joint Multi BERT (Yan et al., 2020), DCGAT achieves better performance especially on parsing. Note that we do not leverage additional syntactic and lexical annotation within a word as Zhang et al. (2014) and Li et al. (2018) do. This demonstrates the advantage of our DCGAT model.

The final results on parsing of the lattice-LSTM BERT end-to-end model are 88.1%, 83.6% and 84.9% on the CTB 5, 6 and 7 datasets, respectively, and the final results of DCGAT are 91.3%, 87.2% and 86.2% on the three datasets, respectively. The improvements are statistically significant at $p < 0.05$ using t-test. Overall, the GAT architecture gives better segmentation, POS-tagging and parsing results compared with the Lattice LSTM model on all datasets, while running much faster. This shows the advantage of using semantic and structural channels for integrating word information, which gives

Model	CTB 5.0			CTB 6.0			CTB 7.0		
	SEG	POS	DEP	SEG	POS	DEP	SEG	POS	DEP
Incremental Joint (Hatori et al., 2012)	96.9	93.0	76.0	96.2	92.0	75.8	96.1	91.3	74.6
Char STD (Zhang et al., 2014)	97.8	94.6	82.1	95.6	91.4	77.1	95.5	90.8	75.7
Char EAG Zhang et al. (2014)	97.8	94.4	82.1	95.7	91.5	77.0	95.5	90.7	75.8
Joint Annotated (Zhang et al., 2015)	98.0	94.5	82.0	-	-	-	-	-	-
NN transition (Kurita et al., 2017)	98.4	94.8	81.4	-	-	-	96.4	91.3	75.3
NN char-level (Li et al., 2018)	96.6	92.9	79.4*	-	-	-	-	-	-
†Joint Multi BERT (Yan et al., 2020)	98.5	-	89.6	-	-	-	97.1	-	85.1
†Lattice-LSTM E2E BERT	98.4	96.2	88.1	97.3	94.6	83.6	96.9	93.9	84.9
†**DCGAT**	**98.7**	**96.5**	**91.3**	**97.3**	**94.8**	**87.2**	**97.3**	**94.4**	**86.2**

Table 6: Final Results. * indicates performance on CTB 5.1. † indicates the BERT pretrained models.

stronger features. Compared with the existing methods for joint Chinese parsing, our final GAT model gives better results on all the three datasets, achieving the best reported accuracies on dependency parsing in the literature.

7.3.2. Comparison with Word Segmentation Methods

According to Table 6, word information brings the most improvements on parsing accuracies, followed by tagging accuracies. Segmentation benefits relatively less. However, as a fully end-to-end model, POS tagging and dependency parsing information can also benefit word segmentation. Thus our joint model can be a competitive choice for the segmentation task alone. We compare the performance of different models with BERT pretraining on the CTB6 word segmentation task in Table 7 (previous work mostly uses the CTB6 version). The first 4 items are the segmentation models trained on the single segmentation task. BiLSTM-CRF-BERT (Gan and Zhang, 2019) uses a BiLSTM network followed by CRF network with pretrained BERT as input. LSAN-CRF (Gan and Zhang, 2019) uses a local self-attention network instead of BiLSTM to incorporate external lexicon. DP-BERT (Huang et al., 2019) and Unified-

Model	F1-score
BiLSTM-CRF-BERT (Gan and Zhang, 2019)	97.2
LSAN-CRF-BERT (Gan and Zhang, 2019)	97.4
DP-BERT (Huang et al., 2019)	97.6
Unified BERT (Ke et al., 2020)	97.2
DCGAT	97.3

Table 7: Word segmentation results based on BERT.

BERT (Ke et al., 2020) are both multi-criteria Chinese word segmentation models, which are trained on a range of extra training sets to enhance the segmenter. DP-BERT (Huang et al., 2019) uses a domain projection for multi-criteria learning while unified BERT (Ke et al., 2020) employs a fully shared model for all criteria.

Our model gives better performances on segmentation than BiLSTM-CRF-BERT and a bit worse result than LSAN-CRF-BERT, which is specially optimised for segmentation. In addition, our model gives comparable results in segmentation compared with DP-BERT and Unified-BERT, which shows the advantage of end-to-end parsing as compared to external segmentation datasets. Compared with Yan et al. (2020), our model can give better segmentation scores.

7.3.3. Comparison with POS-tagging Methods

We compare our DCGAT with other BERT-based models on CTB 5.0 POS-tagging task as shown in Table 8. BERT (Meng et al., 2019) uses a fine-tuned BERT for Chinese POS tagging. Glyce BERT (Meng et al., 2019) combines Glyce information and the BERT model, which designs CNN structures to encode Chinese character images from different styles of historical Chinese scripts. They first separately fine-tune the BERT model and the glyph layer, and finally jointly tune both layers until convergence. We can find that DCGAT outperforms the fine-tuned BERT model, thanks to the use of lexicon information and joint parsing. Compared with Glyce BERT, DCGAT gives a comparable F-score without depending on the glyph layer.

Model	**F-score**
BERT (Meng et al., 2019)	96.1
Glyce BERT (Meng et al., 2019)	96.6
Lattice-LSTM E2E BERT	96.2
DCGAT	96.5

Table 8: POS tagging results on CTB5. All results are based on BERT pretraining.

Task	Character-level Transformer	DCGAT	Diff.
SEG	87.4	88.3	+0.9
POS	81.7	83.8	+2.1
DEP	76.6	80.7	+4.1
DEP′	87.6	91.4	+3.8

Table 9: Recalls of OOV words. DEP′ indicates the parsing recall rate when the dependent word is correctly segmented.

7.4. Analysis

In this section, we focus on DCGAT and discuss the effect of word information.

OOV Table 9 shows the recall of out-of-vocabulary words on the CTB 5.0 test set for different tasks. There are 74.9% OOV words of the CTB 5.0 test set in the lexicon. DCGAT enriched with word information achieves 0.9%, 2.1% and 4.1% absolute improvements over the character-level model on word segmentation, POS tagging and parsing, respectively. It shows that DCGAT benefits from transfer learning and generalization by leveraging external lexicon knowledge. In particular, DCGAT shows significant improvement on parsing. The improvement of parsing results from both better segmentation (+ 0.9%) and better head inference (+3.8%). When given the gold segmentation, DCGCN can give 91.4 F1 scores, while the baseline only achieves 87.6 F1 scores, which shows the external lexicon knowledge is useful for parsing even without considering its effect for word segmentation.

Performance Against the Sentence Length We further analyze the performance

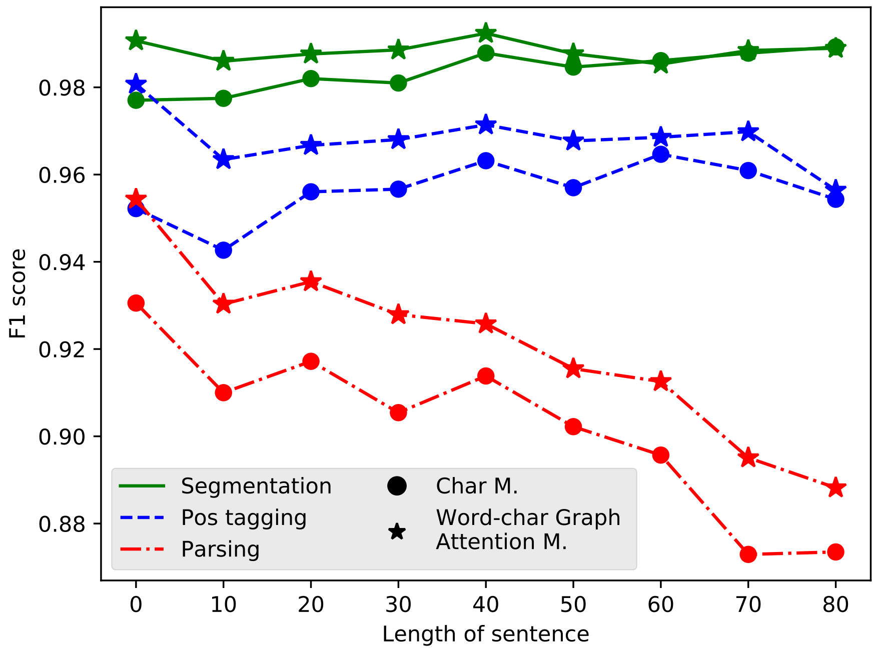

Figure 4: Performance against sentence length.

with respect to different sentence lengths. Figure 4 shows the results. The performance of word segmentation and POS tagging are relatively stable against the sentence length as these two tasks rely more on the local context. There is a tendency of decreasing F1 scores for parsing when the sentence length increases. Compared with the character-level Transformer baseline without using word information, our DCGAT gives about the same improvements on parsing across different sentence lengths. In addition, the use of word information also makes the curve more smooth across different sentence lengths, which shows the benefit of lexicon features in enhancing the model robustness.

Ablation Study We conduct ablation experiments to investigate the effect of different channels of DCGAT, showing three important factors on the CTB 5.0 test set in Table 10. Compared with the full model, removing the structural channel leads to a parsing F1 score decrease by 0.6%. In contrast, excluding the semantic channel decreases the parsing F1 score by 0.8%. Compared with the structural channel, the

Configuration	F1 score	Diff.
DCGAT	91.3	0
- structural channel	90.8	-0.5
- semantic channel	90.6	-0.7
- all word information	90.5	-0.8

Table 10: Ablation test of DCGAT.

semantic channel contains important position information besides semantic similarity, which can be the reason for its better performance. However, the best result is achieved using both channels, which shows that the channels can interact with and complement each other on the final performance. We can also observe that using word information through either channel or both channels combined can improve the parsing performance compared to the character-level attention model ("- all word info" in the table), showing the effectiveness of word information.

Case Study Figure 5 shows sampled outputs of end-to-end Chinese parsing on CTB 5.0. In the first sentence, the head word of token "菲律宾(the Philippines)" is incorrectly recognized as "名字(name)" in the vanilla LSTM model, while DCGAT identifies its correct head "总统(president)". The vanilla LSTM model suffers from polysemous characters "菲(commonly used character in names)", "律(law)" and "宾(commonly used character in names)" . In contrast, DCGAT taking additional word information "菲律宾(the Philippines)" can better understand the sentence and thus gives a correct syntactic parsing result.

In the second sentence, the expression "麦格赛赛奖(Magsaysay Award)" (Gold segmentation: "麦格赛赛(Magsaysay)", "奖(Award)") is incorrectly segmented into tokens "麦格赛(Magsay)" and "赛奖(Competition Award)" by the vanilla LSTM model, which results in errors in the POS tagging and parsing results. It shows that word features can help better identify word boundaries, which is important to POS tagging and parsing. DCGAT segments the sentence correctly, and thus gives the correct result in all three tasks.

Figure 6 shows another case. The token "生前(before one's death)" is a common

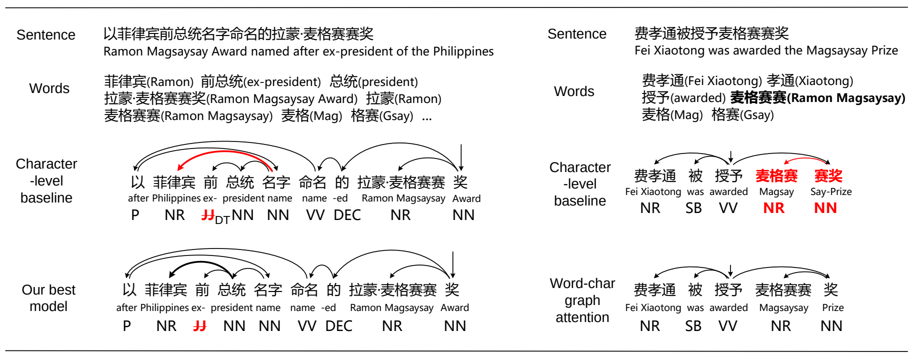
Sentence
以菲律宾前总统名字命名的拉蒙·麦格赛赛奖
Ramon Magsaysay Award named after ex-president of the Philippines
Words
菲律宾(Ramon) 前总统(ex-president) 总统(president)
拉蒙·麦格赛赛奖(Ramon Magsaysay Award) 拉蒙(Ramon)
麦格赛赛(Ramon Magsaysay) 麦格(Mag) 格赛(Gsay) ...
Character -level baseline
以 菲律宾 前 总统 名字 命名 的 拉蒙·麦格赛赛 奖
after Philippines ex- president name name -ed Ramon Magsaysay Award
P NR JJ DT NN NN VV DEC NR NN
Our best model
以 菲律宾 前 总统 名字 命名 的 拉蒙·麦格赛赛 奖
after Philippines ex- president name name -ed Ramon Magsaysay Award
P NR JJ NN NN VV DEC NR NN
Sentence
费孝通被授予麦格赛赛奖
Fei Xiaotong was awarded the Magsaysay Prize
Words
费孝通(Fei Xiaotong) 孝通(Xiaotong)
授予(awarded) 麦格赛赛(Ramon Magsaysay)
麦格(Mag) 格赛(Gsay)
Character -level baseline
费孝通 被 授予 麦格赛 赛奖
Fei Xiaotong was awarded Magsay Say-Prize
NR SB VV NR NN
Word-char graph attention
费孝通 被 授予 麦格赛赛 奖
Fei Xiaotong was awarded Magsaysay Prize
NR SB VV NR NN

Figure 5: Sample Outputs. Errors Are in Red.

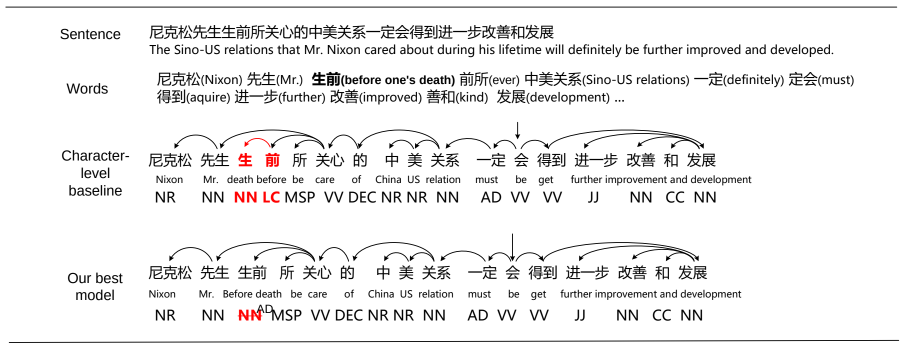
Sentence
尼克松先生生前所关心的中美关系一定会得到进一步改善和发展
The Sino-US relations that Mr. Nixon cared about during his lifetime will definitely be further improved and developed.
Words
尼克松(Nixon) 先生(Mr.) 生前(before one's death) 前所(ever) 中美关系(Sino-US relations) 一定(definitely) 定会(must)
得到(aquire) 进一步(further) 改善(improved) 善和(kind) 发展(development) ...
Character-level baseline
尼克松 先生 生 前 所 关心 的 中 美 关系 一定 会 得到 进一步 改善 和 发展
Nixon Mr. death before be care of China US relation must be get further improvement and development
NR NN NN LC MSP VV DEC NR NR NN AD VV VV JJ NN CC NN
Our best model
尼克松 先生 生前 所 关心 的 中 美 关系 一定 会 得到 进一步 改善 和 发展
Nixon Mr. Before death be care of China US relation must be get further improvement and development
NR NN NN AD MSP VV DEC NR NR NN AD VV VV JJ NN CC NN

Figure 6: Another Case Study. Errors Are in Red.

expression in Chinese and should be regarded as one word. However, the vanilla LSTM model gives the incorrect segmentation result of “生前(before one’s death)”: “生(live)” and “前(before)”. The incorrect segmentation results lead to incorrect word-level dependency parsing (the additional dependency arc “前(before)” → “生(live)”). The DGCGA model gives correct segmentation and parsing result, though the POS tag is still incorrect similar to the vanilla LSTM model. It shows word features may help better identify word boundaries.

In summary, our DCGAT model performs better in identifying dependency heads in addition to word boundaries, thanks to the dual channels for encoding lexicon input.

8. Conclusion

We investigated the effectiveness of lexicon information for graph-based end-to-end Chinese parsing, proposing a novel dual-channel graph attention (DCGAT) encoder to consider both semantic and structural similarity between input characters and words. Compared with LSTM-based representation learning models, our method is more feasible due to better parallelization between characters and convenience to batching. Results on the three datasets show that lexicon information benefits all the three tasks, and our DCGAT model outperforms the lattice-LSTM model, achieving the best results on segmentation, POS tagging and parsing on three Chinese parsing benchmarks in the literature. To our knowledge, we are the first to build an end-to-end Chinese parser exploiting lexicon knowledge.

Acknowledgement

Yue Zhang is the corresponding author. This project is supported by NSFC General Program 61976180 and CCAI-Huawei MindSpore Open Fund CAAIXSJLJJ-2021-046A. We gratefully acknowledge the support of MindSpore, CANN(Compute Architecture for Neural Networks) and Ascend AI Processor used for this research.

References

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M., 2016. Globally normalized transition-based neural networks, in: ACL.

Bohnet, B., 2010. Top accuracy and fast dependency parsing is not a contradiction, in: COLING.

Chen, D., Manning, C., 2014. A fast and accurate dependency parser using neural networks, in: EMNLP.

Chen, X., Shi, Z., Qiu, X., Huang, X., 2017. Dag-based long short-term memory for neural word segmentation. arXiv:1707.00248 .

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding, in: NAACL.

Ding, R., Xie, P., Zhang, X., Lu, W., Li, L., Si, L., 2019. A neural multi-digraph model for Chinese NER with gazetteers, in: ACL.

Dozat, T., Manning, C.D., 2017. Deep biaffine attention for neural dependency parsing. ICML .

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research .

Gamallo, P., Garcia, M., Fernández-Lanza, S., 2012. Dependency-based open information extraction. ROBUS .

Gan, L., Zhang, Y., 2019. Investigating self-attention network for chinese word segmentation. arXiv:1907.11512 .

Gui, T., Zou, Y., Zhang, Q., Peng, M., Fu, J., Wei, Z., Huang, X., 2019. A lexicon-based graph neural network for Chinese NER, 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), Association for Computational Linguistics, Hong Kong, China. pp. 1040–1050. URL: https://aclanthology.org/D19-1096, doi:10.18653/v1/D19-1096.

Guo, Z., Zhang, Y., Lu, W., 2019. Attention guided graph convolutional networks for relation extraction, in: ACL.

Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J., 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. ACL .

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 .

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S., 2019. Parameter-efficient transfer learning for NLP, in: Proceedings of the 36th International Conference on Machine Learning.

Huang, W., Cheng, X., Chen, K., Wang, T., Chu, W., 2019. Toward fast and accurate neural chinese word segmentation with multi-criteria learning. arXiv:1903.04190 .

Ke, Z., Shi, L., Meng, E., Wang, B., Qiu, X., Huang, X., 2020. Unified multi-criteria chinese word segmentation with bert. arXiv:2004.05808 .

Kiperwasser, E., Goldberg, Y., 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. TACL .

Kurita, S., Kawahara, D., Kurohashi, S., 2017. Neural joint model for transition-based chinese syntactic analysis, in: ACL.

Li, H., Zhang, Z., Ju, Y., Zhao, H., 2018. Neural character-level dependency parsing for chinese.

Li, X., Yan, H., Qiu, X., Huang, X., 2020. FLAT: Chinese NER using flat-lattice transformer, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online. pp. 6836–6842. URL: https://aclanthology.org/2020.acl-main.611, doi:10.18653/v1/2020.acl-main.611.

Li, Y., Li, Z., Zhang, M., Wang, R., Li, S., Si, L., 2019. Self-attentive biaffine dependency parsing., in: IJCAI, pp. 5067–5073.

Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., Hovy, E., 2018. Stack-pointer networks for dependency parsing. arXiv:1805.01087 .

Ma, X., Zhao, H., 2012. Fourth-order dependency parsing, in: COLING.

Meng, Y., Wu, W., Wang, F., Li, X., Nie, P., Yin, F., Li, M., Han, Q., Sun, X., Li, J., 2019. Glyce: Glyph-vectors for chinese character representations, in: Advances in Neural Information Processing Systems, pp. 2742–2753.

Michel, P., Levy, O., Neubig, G., 2019. Are sixteen heads really better than one?, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

Miwa, M., Bansal, M., 2016. End-to-end relation extraction using LSTMs on sequences and tree structures, in: ACL.

Poon, H., Domingos, P., 2009. Unsupervised semantic parsing, in: EMNLP.

Song, Y., Shi, S., Li, J., Zhang, H., 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings, in: NAACL: HLT.

Sperber, M., Neubig, G., Pham, N.Q., Waibel, A., 2019. Self-attentional models for lattice inputs, in: ACL.

Strubell, E., Verga, P., Andor, D., Weiss, D., McCallum, A., 2018. Linguistically-informed self-attention for semantic role labeling, in: EMNLP.

Sun, Y., Tang, D., Duan, N., Ji, J., Cao, G., Feng, X., Qin, B., Liu, T., Zhou, M., 2018. Semantic parsing with syntax- and table-aware SQL generation. ACL .

Tang, Z., Wan, B., Yang, L., 2020. Word-character graph convolution network for chinese named entity recognition. IEEE/ACM Transactions on Audio, Speech, and Language Processing 28, 1520–1532.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: NIPS.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2017. Graph attention networks. `arXiv:1710.10903`.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I., 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned, in: Proceedings of the 57th Annual Meeting of the Association for Computational

Linguistics, Association for Computational Linguistics, Florence, Italy. pp. 5797–5808. URL: https://aclanthology.org/P19-1580, doi:`10.18653/v1/P19-1580`.

Wang, W., Chang, B., 2016. Graph-based dependency parsing with bidirectional LSTM, in: ACL.

Wu, L., Zhang, M., 2021. Deep graph-based character-level chinese dependency parsing. IEEE/ACM Transactions on Audio, Speech, and Language Processing 29, 1329–1339. doi:`10.1109/TASLP.2021.3067212`.

Xue, N., Shen, L., 2003. Chinese word segmentation as lmr tagging, in: SIGHAN workshop.

Xue, N., Xia, F., Chiou, F.D., Palmer, M., 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. Natural language engineering .

Yan, H., Qiu, X., Huang, X., 2020. A graph-based model for joint chinese word segmentation and dependency parsing. TACL .

Zhang, H., McDonald, R., 2014. Enforcing structural diversity in cube-pruned dependency parsing, in: ACL.

Zhang, M., Zhang, Y., Che, W., Liu, T., 2013. Chinese parsing exploiting characters.

Zhang, M., Zhang, Y., Che, W., Liu, T., 2014. Character-level chinese dependency parsing, in: ACL.

Zhang, P., Ge, N., Chen, B., Fan, K., 2019. Lattice transformer for speech translation, in: ACL.

Zhang, Y., Li, C., Barzilay, R., Darwish, K., 2015. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing.

Zhang, Y., Qi, P., Manning, C.D., 2018. Graph convolution over pruned dependency trees improves relation extraction, in: EMNLP.

Zhang, Y., Yang, J., 2018. Chinese NER using lattice LSTM, in: ACL.

Zhou, M., 2000. A block-based robust dependency parser for unrestricted Chinese text, in: SIGHAN Workshop.