Pre- and In-Parsing Models for Neural Empty Category Detection

Yufei Chen, Yuanyuan Zhao, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
yufei.chen@pku.edu.cn, zhaoyy1461@gmail.com,
{ws,wanxiaojun}@pku.edu.cn

Abstract

Motivated by the positive impact of empty categories on syntactic parsing, we study neural models for pre- and in-parsing detection of empty categories, which has not previously been investigated. We find several non-obvious facts: (a) BiLSTM can capture non-local contextual information which is essential for detecting empty categories, (b) even with a BiLSTM, syntactic information is still able to enhance the detection, and (c) automatic detection of empty categories improves parsing quality for overt words. Our neural ECD models outperform the prior state-of-the-art by significant margins.

1 Introduction

Encoding unpronounced nominal elements, such as dropped pronouns and traces of dislocated elements, the empty category is an important piece of machinery in representing the (deep) syntactic structure of a sentence (Carnie, 2012). Figure 1 shows an example. In linguistic theory, e.g. Government and Binding (GB; Chomsky, 1981), empty category is a key concept bridging S-Structure and D-Structure, due to its possible contribution to trace *movements*. In practical treebanking, empty categories have been used to indicate long-distance dependencies, discontinuous constituents, and certain dropped elements (Marcus et al., 1993; Xue et al., 2005). Recently, there has been an increasing interest in automatic empty category detection (ECD; Johnson, 2002; Seeker et al., 2012; Xue and Yang, 2013; Wang et al., 2015). And it has been shown that ECD is able to improve the linear model-based dependency parsing (Zhang et al., 2017b).

There are two key dimensions of approaches for ECD: the relationship with parsing and statistical disambiguation. Considering the relationship with parsing, we can divide ECD models into three types: (1) Pre-parsing approach (e.g. Dienes and Dubey (2003)) where empty categories are identified without using syntactic analysis, (2) In-parsing approach (e.g. Cai et al. (2011); Zhang et al. (2017b)) where detection is integrated into a parsing model, and (3) Post-parsing approach (e.g. Johnson (2002); Wang et al. (2015)) where parser outputs are utilized as clues to determine the existence of empty categories. For disambiguation, while early work on dependency parsing focused on linear models, recent work started exploring deep learning techniques for the post-parsing approach (Wang et al., 2015). From the above two dimensions, we show all existing systems for ECD in Table 1. Neural models for pre- and in-parsing ECD have not been studied yet. In this paper, we fill this gap in the literature.

	Pre-Parsing	In-Parsing	Post-Parsing
Linear	✔	✔	✔
Neural	✘	✘	✔

Table 1: ECD approaches that have been investigated.

It is obvious that empty categories are highly related to surface syntactic analysis. To determine the existence of empty elements between two overt words relies on not only the *sequential contexts* but also the *hierarchical contexts*. Traditional linear structured prediction models, e.g. conditional random fields (CRF), for sequence structures are rather weak to capture hierarchical contextual information which is essentially *non-local* for their architectures. Accordingly, pre-parsing models based on linear disambiguation techniques fail to produce comparable accuracy to the other two models. In striking contrast, RNN based se-

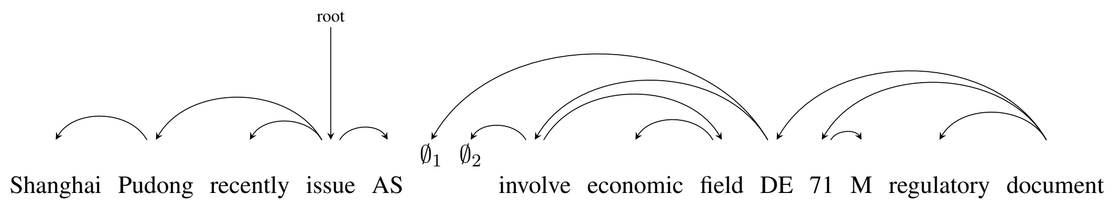


Figure 1: An example from CTB: *Shanghai Pudong recently enacted 71 regulatory documents involving the economic fields*. The dependency structure is according to Xue (2007). "$\emptyset_1$" indicates a null operator that represents empty relative pronouns. "$\emptyset_2$" indicates a trace left by relativization.

quence labeling models have been shown very powerful to capture non-local information, and therefore have great potential to advance the pre-parsing approach for ECD. In this paper, we propose a new bidirectional LSTM (BiLSTM) model for pre-parsing ECD using information about contextual words.

Previous studies highlight the usefulness of syntactic analysis for ECD. Furthermore, syntactic parsing of overt words can benefit from detection of empty elements and vice versa (Zhang et al., 2017b). In this paper, we follow Zhang et al.'s encouraging results obtained with linear models and study first- and second-order neural models for in-parsing ECD. The main challenge for neural in-parsing ECD is to encode empty element candidates and integrate the corresponding embeddings into a parsing model. We focus on the state-of-the-art parsing architecture developed by Kiperwasser and Goldberg (2016) and Dozat and Manning (2016), which use BiLSTMs to extract features from contexts followed by a nonlinear transformation to perform local scoring.

To evaluate the effectiveness of deep learning techniques for ECD, we conduct experiments on a pro-drop language, i.e. Chinese. The empirical evaluation indicates some non-obvious facts:

1. Neural ECD models outperform the prior state-of-the-art by significant margins. Even a pre-parsing model without any syntactic information outperforms the best existing linear in-parsing and post-parsing ECD models.

2. Incorporating empty elements can help neural dependency parsing. This parallels Zhang et al.'s investigation on linear models.

3. Our in-parsing neural models obtain better predictions than the pre-parsing model.

The implementation of all models is available at `https://github.com/draplater/empty-parser`.

2 Pre-Parsing Detection

2.1 Context of Empty Categories

Sequential Context Perhaps, it is the most intuitive idea to view a natural language sentence as a word-by-word sequence. Analyzing contextual information by modeling neighboring words according to this sequential structure is a very basic view for dealing with a large number of NLP tasks, e.g. POS tagging and syntactic parsing. It is also important to consider sequential contexts for ECD to derive the horizontal features that exploit the lexical context of the current pending point, presented as one or more preceding and following word tokens, as well as their part-of-speech tags (POS).

Hierarchical Context The detection of ECs requires broad contextual knowledge. Besides one-dimensional representation, vertical features are equally essential to express the empty element. The hierarchical structure is a compact reflection of the syntactic content. By integrating the hierarchical context, we can analyze the regular distributional pattern of ECs in a syntactic tree. More specifically, it means considering the head information of the EC and relevant dependencies to augment the prediction.

Both *sequential* and *hierarchical* contexts are essential to determine the existence of empty elements between two overt words. Even words close to each other in a hierarchical structure may appear far apart in sequential representations, which makes it hard for linear sequential tagging models to catch the hierarchical contextual information. RNN based sequence models have been proven very powerful to capture non-local features. In this paper, we show that LSTM is able to advance the pre-parsing ECD significantly.

Interspace:	@@	(issue)	@@	(AS)	@@	(involve)	@@	(economic)
	O	VV	O	AS	*OP**T*	VV	O	NN
Pre2 and *Pre3*:		(issue)		(AS)		(involve)		(economic)
		VV		AS		VV#pre1=*T*#pre2=*OP*		NN
Prepost:		(issue)		(AS)		(involve)		(economic)
		VV		AS#post=*OP*		VV#pre1=*T*		NN

Figure 2: An example of four kinds of annotations. The phrase is cut out from the sentence in Figure 1. "@@" means interspaces between words.

2.2 A Sequence-Oriented Model

In the sequence-oriented model, we formulate ECD as a sequence labeling problem. In general, we attach ECs to surrounding overt tokens to represent their identifications, i.e. their locations and types. We explore four sets of annotation specifications, denoted as *Interspace*, *Pre2*, *Pre3* and *Prepost*, respectively. Following is the detailed descriptions.

Interspace We convert ECs' information into different tags of the interspaces between words. The assigned tag is the concatenation of ECs between the two words. If there is no EC, we just tag the interspace as *O*. Specially, according to our observation that only one EC occurs at the end of the sentence in our data set, we simply count on the heading space of sentences instead of the one standing at the end. Assume that there are n words in a given sentence, then there will be $2 * n$ items (n words and n interspaces) to tag.

Pre2 and Pre3 We stick ECs to words following them. In experiments using POS information, ECs are attached to the POS of the next word, while the normal words are just tagged with their POS. In experiments without POS information, ECs are straightly regarded as the label of the following words. Words without ECs ahead are consistently tagged using an empty marker. Similar to *Interspace*, linearly consecutive ECs are concatenated as a whole. *Pre2* means that at most two preceding consecutive ECs are considered while *Pre3* limits the considered continuous length to three. The determination of window lengths are grounded in the distribution of ECs' continuous lengths as shown in Table 2.

Prepost Considering that it may be a challenge to capture long-distance features, we introduce another labeling rule called *Prepost*. Different from *Pre2* and *Pre3*, the responsibility for presenting ECs will be shared by both the preceding and the following words. Whereas, tags heading sentences will remain unchanged. Particularly, if the amount of consecutive ECs in the current position is an odd number, we choose to attach the extra EC to the following word for consistency and clarity.

	1	2	3	4
Train	7499	3702	142	5
Dev	530	233	10	0
Test	900	433	19	0

Table 2: The distribution of ECs' continuous lengths in training, development and test data.

Take part of the sentence in Figure 1 as an example. As described above, the four kinds of representations are depicted in Figure 2. To investigate the effect of POS in the tagging process, we also conduct experiments by integrating POS to the tagging process. For *Interspace*, POS tags are individual output labels, while for other representations, the POS information is used to divide an empty category integrated tag into subtypes.

2.3 Tagging Based on LSTM-CRF

In order to capture long-range syntactic information for accurate disambiguation in pre-parsing phase, we build a LSTM-CRF model inspired by the neural network proposed in Ma and Hovy (2016). A BiLSTM layer is set up on character embeddings for extracting character-level representations of each word, which is concatenated with the pre-trained word embedding before feeding into another BiLSTM layer to capture contextual information. Thus we have obtained dense and continuous representations of the words in given sentences. The last part is to decode with linear chain CRF which can optimize the output sequence by factoring in local characteristics. Dropout layers both before and after the sentence-level network serve to prevent over-fitting.

3 In-Parsing Detection

Zhang et al. (2017b) designs novel algorithms to produce dependency trees in which empty elements are allowed. Their results show that integrating empty categories can augment the parsing of overt tokens when structured perceptron, a global linear model, is applied for disambiguation. From a different perspective, by jointing ECD and dependency parsing, we can utilize full syntactic information in the process of detecting ECs. Parallel to their work, we explore the effect of ECD on the neural dependency based parsing in this section.

3.1 Joint ECD and Dependency Parsing

To perform ECD and dependency parsing in a unified framework, we formulate the issue as an optimization problem. Assume that we are given a sentence s with n normal words. We use an index set $\mathcal{I}_o = \{(i,j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible overt dependency edges, and use $\mathcal{I}_c = \{(i,\phi_j)|i,j \in \{1,\cdots,n\}\}$ to denote all possible covert dependency edges. ϕ_j denotes an empty node that precede the jth word. Then a dependency parse with empty nodes can be represented as a vector:

$$\boldsymbol{z} = \{z(i,j) : (i,j) \in \mathcal{I}_o \cup \mathcal{I}_c\}.$$

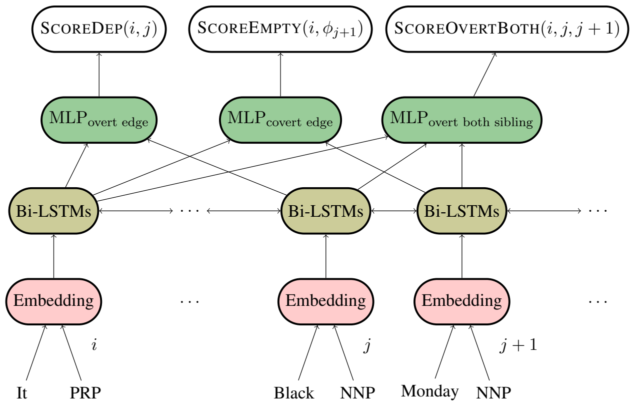
Let $\mathcal{Z}$ denote the set of all possible $\boldsymbol{z}$, and $\textsc{Part}(\boldsymbol{z})$ denote the factors in the dependency tree, including edges (and edge siblings in the second-order model). Then parsing with ECD can be defined as a search for the highest-scored $\boldsymbol{z}^*(s)$ in all compatible analyses, just like parsing without empty elements:

$$\begin{aligned}\boldsymbol{z}^*(s) &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(s)} \textsc{Score}(s,\boldsymbol{z}) \\ &= \arg\max_{\boldsymbol{z}\in\mathcal{Z}(s)} \sum_{p\in\textsc{Part}(\boldsymbol{z})} \textsc{ScorePart}(s,p)\end{aligned}$$

The graph-based parsing algorithms proposed by Zhang et al. are based on two properties: ECs can only serve as dependents and the number of successive ECs is limited. The latter trait makes it reasonable to treat consecutive ECs governed by the same head as one *word*. We also follow this set-up.

3.2 Scoring Based on BiLSTM

Kiperwasser and Goldberg (2016) proposed a simple yet effective architecture to implement neural



$\textsc{ScoreDep}(i,j)$
$\textsc{ScoreEmpty}(i,\phi_{j+1})$
$\textsc{ScoreOvertBoth}(i,j,j+1)$
$\text{MLP}_{\text{overt edge}}$
$\text{MLP}_{\text{covert edge}}$
$\text{MLP}_{\text{overt both sibling}}$
Bi-LSTMs
Bi-LSTMs
Bi-LSTMs
Embedding
Embedding
Embedding
i
j
$j+1$
It
PRP
Black
NNP
Monday
NNP

Figure 3: The neural network structure when parsing sentence "It wasn't Black Monday." 5 MLPs is used for overt edges (i,j), covert edges (i,ϕ_j), overt-both siblings (i,j,k), covert-inside siblings (i,ϕ_j,k) and covert-outside siblings (i,j,ϕ_k) respectively, and 3 of them are shown in the graph.

dependency parsers. In particular, a BiLSTM is utilized as a powerful *feature extractor* to assist a dependency parser. Mainstream data-driven dependency parsers, including both transition- and graph-based ones, can apply useful word vectors provided by a BiLSTM to conduct the disambiguation. Following Kiperwasser and Goldberg (2016)'s experience on graph-based dependency parser, we implement such a parser to recover empty categories and to evaluate the impact of empty categories on surface parsing.

Here we present details of the design of our parser. A vector is associated with each word or POS-tag to transform them into continuous and dense representations. We use pre-trained word embeddings and random initialized POS-tag embeddings.

The concatenation of the word embedding and the POS-tag embedding of each word in a specific sentence is used as the input of BiLSTMs to extract context related feature vectors r_i.

$$\boldsymbol{r}_{1:\text{n}} = \text{BiLSTM}(s; 1:n)$$

The context related feature vectors are fed into a non-linear transformation to perform scoring.

3.3 A First-Order Model

In the first-order model, we only consider the head and the dependent of the possible dependency arc. The two feature vectors of each word pair is scored with a non-linear transformation g as the first-order score. When words i and j are overt words, we define the score function in sentence s as follows,

$$\begin{aligned}&\textsc{ScoreDep}(s, i, j)\\&= \boldsymbol{W}_2 \cdot g(\boldsymbol{W}_{1,1} \cdot \boldsymbol{r}_i + \boldsymbol{W}_{1,2} \cdot \boldsymbol{r}_j + \boldsymbol{b})\end{aligned}$$

$\boldsymbol{W}_2$, $\boldsymbol{W}_{1,1}$ and $\boldsymbol{W}_{1,2}$ denote the weight matrices in linear transformations. The score of covert edge from word i to word ϕ_j is calculated in a similar way with different parameters:

$$\begin{aligned}&\textsc{ScoreEmpty}(s, i, \phi_j)\\&= \boldsymbol{W}_2' \cdot g(\boldsymbol{W}_{1,1}' \cdot \boldsymbol{r}_i + \boldsymbol{W}_{1,2}' \cdot \boldsymbol{r}_j + \boldsymbol{b}')\end{aligned}$$

These non-linear transformations are also known as Multiple Layer Perceptrons(MLPs). The total score in our first-order model is defined as follows,

$$\begin{aligned}\textsc{Score}(s, \boldsymbol{z}) = &\sum_{(i,j)\in\textsc{Dep}(\boldsymbol{z})} \textsc{ScoreDep}(s, i, j)\\&+ \sum_{(i,\phi_j)\in\textsc{DepEmpty}(\boldsymbol{z})} \textsc{ScoreEmpty}(s, i, \phi_j)\end{aligned}$$

$\textsc{Dep}(\boldsymbol{z})$ and $\textsc{DepEmpty}(\boldsymbol{z})$ denote all overt and covert edges in $\boldsymbol{z}$ respectively. Because each overt and covert edge is selected independently of the others, the decoding process can be seen as calculating the maximum subtree from overt edges(we use Eisner Algorithm in our experiments) and appending each covert edge (i, ϕ_j) when $\textsc{ScoreEmpty}(i, \phi_j) > 0$.

3.4 A Second-Order Model

In the second-order model, we also consider sibling arcs. We extend the neural network in section 3.3 to perform the second-order parsing. We calculate second-order scores(scores defined over sibling arcs) in a similar way. Each pair of overt sibling arcs, for example, (i, j) and (i, k) $(j < k)$, is denoted as (i, j, k) and scored with a non-linear transformation.

$$\begin{aligned}&\textsc{ScoreOvertBoth}(s, i, j, k) =\\&\boldsymbol{W}_2'' \cdot g(\boldsymbol{W}_{1,1}'' \cdot \boldsymbol{r}_i + \boldsymbol{W}_{1,2}'' \cdot \boldsymbol{r}_j + \boldsymbol{W}_{1,3}'' \cdot \boldsymbol{r}_k + \boldsymbol{b}'')\end{aligned}$$

Zhang et al. (2017b) defines two kinds of second-order scores to describe the interaction between concrete nodes and empty categories: the covert-inside sibling (i, ϕ_j, k) and covert-outside sibling (i, j, ϕ_k). Their scores can be calculated in a similar way with different parameters.

And finally, the score function over the whole syntactic analysis is defined as:

$$\begin{aligned}\textsc{Score}(s, \boldsymbol{z}) = &\sum_{(i,j)\in\textsc{Dep}(\boldsymbol{z})} \textsc{ScoreDep}(s, i, j)\\&+ \sum_{(i,\phi_j)\in\textsc{DepEmpty}(\boldsymbol{z})} \textsc{ScoreEmpty}(s, i, \phi_j)\\&+ \sum_{(i,j,k)\in\textsc{OvertBoth}(\boldsymbol{z})} \textsc{ScoreOvertBoth}(s, i, j, k)\\&+ \sum_{(i,\phi_j,k)\in\textsc{CovertIn}(\boldsymbol{z})} \textsc{ScoreCovertIn}(s, i, \phi_j, k)\\&+ \sum_{(i,j,\phi_k)\in\textsc{CovertOut}(\boldsymbol{z})} \textsc{ScoreCovertOut}(s, i, j, \phi_k)\end{aligned}$$

$\textsc{OvertBoth}(\boldsymbol{z})$, $\textsc{CovertIn}(\boldsymbol{z})$ and $\textsc{CovertOut}(\boldsymbol{z})$ denotes overt-both, covert-inside and covert-outside siblings of $\boldsymbol{z}$ respectively. Totally 5 MLPs are used to calculate the 5 types of scores. The network structure is shown in Figure 3.

Labeled Parsing Similar to Kiperwasser and Goldberg (2016) and Zhang et al. (2017a), we use a two-step process to perform labeled parsing: conduct an unlabeled parsing and assign labels to each dependency edge. The labels are determined with the nonlinear classification. We use different nonlinear classifiers for edges between concrete nodes and empty categories.

Training In order to update graphs which have high model scores but are very wrong, we use a margin-based approach to compute loss from the gold tree T^* and the best prediction $\hat{T}$ under the current model.

We define the *loss* term as:

$$\max(0, \Delta(T^*, \hat{T}) - \textsc{Score}(T^*) + \textsc{Score}(\hat{T}))$$

The margin objective Δ measures the similarity between the gold tree T^* and the prediction $\hat{T}$. Following Kiperwasser and Goldberg (2016)'s experience of loss augmented inference, we define Δ as the count of dependency edges in prediction results but not belonging to the gold tree.

3.5 Structure Regularization

ECD significantly increases the search space for parsing. This results in a side effect for practical parsing. Given the limit of available annotations for training, searching for more complex structures in a larger space is harmful to the generalization ability in structured prediction (Sun, 2014). To control structure-based overfitting, we train a normal dependency parser, namely parser for overt words only, and use its first- and second-order scores to augment the corresponding score functions in the joint parsing and ECD model. At the training phase, the two parsers are trained separately, while at the test phase, the scores are calculated by individual models and added for decoding.

4 Experiments

4.1 Experimental Setup

4.1.1 Data

We conduct experiments on a subset of Penn Chinese Treebank (CTB; Xue et al., 2005) 9.0. As a pro-drop language, the empty category is a very useful method for representing the (deep) syntactic analysis in Chinese language. Empty categories in CTB is divided into six classes: *pro*, *PRO*, *OP*, *T*, *RNR* and *, which were described in detail in Xue and Yang (2013); Wang et al. (2015). For comparability with the state-of-the-art, the division of training, development and testing data is coincident with the previous work (Xue and Yang, 2013).

Our experiments can be divided into two groups. The first group is conducted on the linear conditional random field (Linear-CRF) model and LSTM-CRF tagging model to evaluate gains from the introduction of neural structures. The second group is designed for the dependency-based in-parsing models.

4.1.2 Evaluation Metrics

We adopt two kinds of metrics for the evaluation of our experiments. The first one focuses on EC's position and type, in accordance with the *labeled empty elements* measure proposed by Cai et al. (2011), which can be implemented on all models in our experiments. The second one is stricter. Besides position and type, it also checks EC's head information. An EC is considered to be correct, only when all the three parts are the same as the corresponding gold standard. Thus only models involved in dependency structures can be evaluated according to the latter metric. Based on above measures of the two degrees, we evaluate our neural pre- and in-parsing models regarding each type of EC as well as overall performance.

Besides, to compare different models' abilities to capture non-local information, we design *Dependency Distance* to indicate the number of words from one EC to its head, not counting other ECs on the path. Taking the two ECs in Figure 1 as an example, $\emptyset_2$ has a *Dependency Distance* of 0 while $\emptyset_1$'s *Dependency Distance* is 3. We calculate labeled recall scores for enumerated *Dependency Distance*. A higher score means greater capability to catch and to represent long-distance details.

4.2 Results of Pre-Parsing Models

Table 3 shows overall performances of the two sequential models on development data. From the results, we can clearly see that the introduction of neural structure pushes up the scores exceptionally. The reason is that our LSTM-CRF model not only benefits from the linear weighted combination of local characteristics like ordinary CRF models, but also has the ability to integrate more contextual information, especially long-distance information. It confirms LSTM-based models' great superiority in sequence labeling problems.

Further more, we find that the difference among the four kinds of representations is not so obvious. The most performing one with LSTM-CRF model is *Interspace*, but the advantage is narrow. *Pre3* uses a larger window length to incorporate richer contextual tokens, but at the same time, the searching space for decoding grows larger. It explains that the performance drops slightly with increasing window length. In general, experiments with POS tags show higher scores as more syntactic clues are incorporated.

We compare LSTM-CRF with other state-of-the-art systems in Table 4[1]. We can see that a simple neural pre-parsing model outperforms state-of-the-art linear in-parsing systems. Analysis about results on different EC types as displayed in Table 5 shows that the sequence-oriented pre-parsing model is good at detecting *pro* compared with previous systems, which is used widely in pro-drop languages. Additionally, the model succeeds in detecting seven * EC tokens in evaluating process. * indicates trace left by passivization as well as raising, and is very rare in training data. Previous models usually cannot identify any *. This detail reflects that the LSTM-CRF model can make the

[1] Wang et al. (2015) reported an overall F-score of 71.7. But their result is based on the gold standard syntactic analysis.

	Linear CRF						LSTM-CRF					
	Without POS			With POS			Without POS			With POS		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Interspace	74.6	20.6	32.2	71.2	30.3	42.5	67.9	59.8	63.6	73.0	61.6	66.8
Pre2	72.4	30.1	42.5	72.8	32.4	44.8	71.1	58.3	64.1	74.8	57.4	65.0
Pre3	73.1	30.2	42.8	73.0	32.5	44.9	71.1	58.5	64.2	73.8	57.0	64.3
Prepost	70.9	32.9	45.0	74.4	30.3	43.1	71.0	57.6	63.6	72.9	58.6	65.0

Table 3: The overall performance of the two sequential models on development data.

	P	R	F_1
Pre-parsing	67.3	54.7	60.4
In-parsing	72.6	55.5	62.9
In-parsing*	70.9	54.1	61.4
(Xue and Yang, 2013)*	65.3	51.2	57.4
(Cai et al., 2011)	66.0	54.5	58.6

Table 4: The overall performance on test data. "*" indicates more stringent evaluation metrics.

EC Type	Total	Correct	P	R	F_1
pro	315	85	52.5	27.0	35.6
PRO	300	183	58.8	61.0	59.9
OP	575	338	73.0	58.8	65.1
T	580	355	73.3	61.2	66.7
RNR	34	30	62.5	88.2	73.2
*	19	7	46.7	36.8	41.2
Overall	1823	998	67.3	54.7	60.4

Table 5: Occurrences of different ECs in test data and detailed results of *Interspace* with POS information.

most of limited training data compared with existing systems.

4.3 Results of In-Parsing Models

Table 6 presents detailed results of the in-parsing models on test data. Compared with the state-of-the-art, the first-order model performs a little worse while the second-order model achieves a remarkable score. The first-order parsing model only constrains the dependencies of both the covert and overt tokens to make up a tree. Due to the loose scoring constraint of the first-order model, the prediction of empty nodes is affected little from the prediction of dependencies of overt words. The four bold numbers in the table intuitively elicits the conclusion that integrating an *empty* edge and its sibling *overt edges* is necessary to boost the performance. It makes sense because empty categories are highly related to syntactic analysis. When we conduct ECD and dependency parsing simultaneously, we can leverage more hierarchical contextual information. Comparing results regarding EC types, we can find that *OP* and *T* benefit most from the parsing information, the F_1 score increasing by about ten points, more markedly than other types.

	First-order			Second-order		
Type	P	R	F_1	P	R	F_1
pro	52.5	16.8	25.5	54.4	19.7	28.9
PRO	59.7	47.3	52.8	60.6	58.0	59.3
OP	74.5	55.8	63.8	79.6	67.8	73.2
T	70.6	51.7	59.7	77.3	62.8	69.3
RNR	70.8	50.0	58.6	77.8	61.8	68.9
*	0.0	0.0	0.0	0.0	0.0	0.0
Overall	68.2	45.7	**54.7**	72.6	55.5	**62.9**
Evaluation with Head						
pro	50.5	16.2	24.5	52.6	19.1	28.0
PRO	58.4	46.3	51.7	57.8	55.3	56.6
OP	72.2	54.1	61.8	78.6	67.0	72.3
T	68.5	50.2	57.9	75.4	61.2	67.6
RNR	70.8	50.0	58.6	77.8	61.8	68.9
*	0.0	0.0	0.0	0.0	0.0	0.0
Overall	66.3	44.4	**53.2**	70.9	54.1	**61.4**

Table 6: The performances of the first- and second-order in-parsing models on test data.

4.4 Results on Dependency Parsing

Table 7 shows the impact of automatic detection of empty categories on parsing overt words. We compare the results of both steps in labeled parsing. We can clearly see that integrating empty elements into dependency parsing can improve the neural parsing accuracy of overt words. Besides, when jointing parsing models both without and with ECs together, we can push up the performance further. These results confirm the conclusion in Zhang et al. (2017b) that empty elements

	-EC	+EC	-+EC
Unlabeled	87.6	88.9	89.6
Labeled	84.6	85.9	86.6

Table 7: Accuracies of both unlabeled and labeled parsing on development data. -EC indicates parsing without empty categories. +EC indicates the second-order in-parsing models. -+EC indicates jointing parsing models both without and with ECs together.

help parse the overt words. The main reason lies in that the existence of ECs provides extra structural information which can reduce approximation errors in a structured prediction problem.

According to above analysis, we can draw a conclusion that ECD and syntactic parsing can promote each other mutually. That partially explains why in-parsing models can outperform pre-parsing models. Meanwhile, it provides a new approach to improving the dependency parsing quality in a unified framework.

4.5 Impact of Dependency Distance

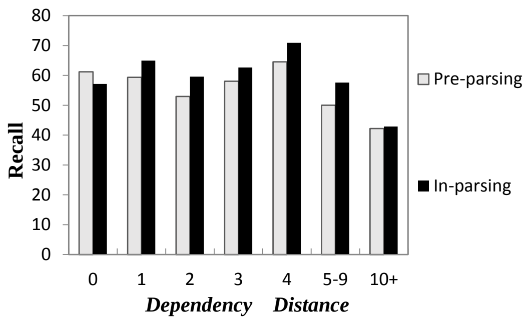


Figure 4: Recall scores of different models regarding *Dependency Distance*. ”Pre-parsing” and ”In-parsing” refer to the LSTM-CRF model and the dependency-based in-parsing model respectively.

We compare pre- and in-parsing models regarding *Dependency Distance*. The former refers to the LSTM-CRF model while the latter means the dependency-based in-parsing model. Figure 4 shows the results. The abscissa value ranges from 0 to 26, with the longest dependency arc spanning 26 non-EC word tokens. We can see that long-distance disambiguation is a challenge shared by both models. When the value of *Dependency Distance* exceeds four, the recall score drops gradually with abscissa increasing. Based on the comparison of two sets of data, we can find that in-parsing model performs better on ECs which are close to their heads. However, as for ECs which are far apart from their heads, two models have performed almost exactly alike. It demonstrates that LSTM structure is capable of capturing non-local features, making up for no exposure to parsing information.

4.6 Challenges

On the whole, the most challenging EC type is *pro*. We assume that it is because that pro-drop situations are complicated and diverse in Chinese language. According to Chinese linguistic theory, pronouns are dropped as a result of continuing from the preceding discourse or just idiomatic rules, such as the ellipsis of the first person pronoun “/I” in the subject position. To fill this gap, we may need to extract more deep structural features.

Another difficulty is the detection of consecutive ECs. In the result of our experiments, in-parsing dependency-based model can only accurately detect up to two consecutive ECs. Too many empty elements in the same sentence conceal too much syntactic information, making it hard to disclose the original structure.

Moreover, in view of the fact that ECs play an essential role in syntactic analysis, the current detection accuracy of ECs is far from enough. We still have a long way to go.

5 Related Work

The detection of empty categories is an essential ground for many downstream tasks. For example, Chung and Gildea (2010) has proved that automatic empty category detection has a positive impact on machine translation. Zhang et al. (2017b) shows that ECD can benefit linear syntactic parsing of overt words. To accurately distinguish empty elements in sentences, there are generally three approaches. The first method is to build pre-processors before syntactic parsing. Dienes and Dubey (2003) proposed a shallow *trace tagger* which can detect discontinuities. And it can be combined with unlexicalized PCFG parsers to implement deep syntactic processing. Due to the lack of phrase structure information, it did not acquire remarkable results. The second method is to integrate ECD into parsing, as shown in

Schmid (2006) and Cai et al. (2011), which involved empty elements in the process of generating parse trees. Another in-parsing system is proposed in Zhang et al. (2017b). Zhang et al. (2017b) designed algorithms to produce dependency trees in which empty elements are allowed. To add empty elements into dependency structures, they extend Eisner's first-order DP algorithm for parsing to second- and third-order algorithms. The last approach to recognizing empty elements is post-parsing methods. Johnson (2002) proposed a simple pattern-matching algorithm for recovering empty nodes in phrase structure trees while Campbell (2004) presented a rule-based algorithm. Xue and Yang (2013) conducted ECD based on dependency trees. Their methods can leverage richer syntactic information, thus have achieved more satisfying scores.

As neural networks have been demonstrated to have a great ability to capture complex features, it has been applied in multiple NLP tasks (Bengio and Schwenk, 2006; Collobert et al., 2011). Neural methods have also explored in distinguishing empty elements. For example, Wang et al. (2015) described a novel ECD solution using distributed word representations and achieved the state-of-the-art performance. Based on above work, we explore neural pre- and in-parsing models for ECD.

6 Conclusion

Neural networks have played a big role in multiple NLP tasks recently owing to its nonlinear mapping ability and the avoidance of human-engineered features. It should be a well-justified solution to identify empty categories as well as to integrate empty categories into syntactic analysis. In this paper, we study neural models to detect empty categories. We observe three facts: (1) BiLSTM significantly advances the pre-parsing ECD. (2) Automatic ECD improves the neural dependency parsing quality for overt words. (3) Even with a BiLSTM, syntactic information can enhance the detection further. Experiments on Chinese language show that our neural model for ECD exceptionally boosts the state-of-the-art detection accuracy.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61772036, 61331011) and the Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Weiwei Sun is the corresponding author.

References

Yoshua Bengio and Holger Schwenk. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*. Springer, page 137186.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 212–216. http://www.aclweb.org/anthology/P11-2037.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 645–652. https://doi.org/10.3115/1218955.1219037.

A. Carnie. 2012. *Syntax: A Generative Introduction 3rd Edition and The Syntax Workbook Set*. Introducing Linguistics. Wiley. https://books.google.com/books?id=jhGKMAEACAAJ.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris Publications, Dordecht.

Tagyoung Chung and Daniel Gildea. 2010. Effects of empty categories on machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 636–645. http://www.aclweb.org/anthology/D10-1062.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(1):2493–2537.

Pétr Dienes and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sapporo, Japan, pages 431–438. https://doi.org/10.3115/1075096.1075151.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 136–143. https://doi.org/10.3115/1073083.1073107.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized pcfgs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 177–184. https://doi.org/10.3115/1220175.1220198.

Wolfgang Seeker, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1081–1090. http://www.aclweb.org/anthology/C12-2105.

Xu Sun. 2014. Structure regularization for structured prediction. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2402–2410. http://papers.nips.cc/paper/5563-structure-regularization-for-structured-prediction.pdf.

Xun Wang, Katsuhito Sudoh, and Masaaki Nagata. 2015. Empty category detection with joint context-label embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 263–271. http://www.aclweb.org/anthology/N15-1030.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11:207–238. https://doi.org/10.1017/S135132490400364X.

Nianwen Xue. 2007. Tapping the implicit information for the PS to DS conversion of the Chinese treebank. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistics Theories*.

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1051–1060. http://www.aclweb.org/anthology/N13-1125.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017a. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 665–676. http://www.aclweb.org/anthology/E17-1063.

Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2017b. The covert helps parse the overt. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 343–353. http://aclweb.org/anthology/K17-1035.