Hierarchical Pointer Net Parsing

Linlin Liu[12*†], **Xiang Lin**[1†], **Shafiq Joty**[13], **Simeng Han**[1], **Lidong Bing**[2]
[1]Nanyang Technological University, Singapore
[2]R&D Center Singapore, Machine Intelligence Technology, Alibaba DAMO Academy
[3]Salesforce Research Asia, Singapore
{linx0057, srjoty, hans0035}@ntu.edu.sg
{linlin.liu, l.bing}@alibaba-inc.com

Abstract

Transition-based top-down parsing with pointer networks has achieved state-of-the-art results in multiple parsing tasks, while having a linear time complexity. However, the decoder of these parsers has a sequential structure, which does not yield the most appropriate inductive bias for deriving tree structures. In this paper, we propose hierarchical pointer network parsers, and apply them to dependency and sentence-level discourse parsing tasks. Our results on standard benchmark datasets demonstrate the effectiveness of our approach, outperforming existing methods and setting a new state-of-the-art.

1 Introduction

Parsing of sentences is a core natural language understanding task, where the goal is to construct a tree structure that best describes the relationships between the tree constituents (*e.g.,* words, phrases). For example, Figure 1 shows examples of a **dependency tree** and a sentence-level **discourse tree** that respectively represent how the words and clauses are related in a sentence. Such parse trees are directly useful in numerous NLP applications, and also serve as intermediate representations for further language processing tasks such as semantic and discourse processing.

Existing approaches to parsing can be distinguished based on whether they employ a greedy **transition-based** algorithm (Marcu, 1999; Zhang and Nivre, 2011; Wang et al., 2017) or a globally optimized algorithm such as **graph-based** methods for dependency parsing (Eisner, 1996) or **chart parsing** for discourse (Soricut and Marcu, 2003; Joty et al., 2015). Transition-based parsers build the tree incrementally by making a series of shift-reduce decisions. The advantage of this method is that the parsing time is linear with respect to the sequence length. The limitation, however, is that the decisions made at each step are based on local information, disallowing the model to capture long distance dependencies and also causing error propagation to subsequent steps. Recent methods attempt to address this issue using neural networks capable of remembering long range relationships such as Stacked LSTMs (Dyer et al., 2015; Ballesteros et al., 2015) or using globally normalized models (Andor et al., 2016).

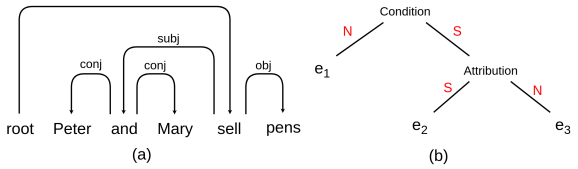


Figure 1: (a) A dependency tree for a sentence; (b) a discourse tree for the sentence "[Now that's name-dropping,]$_{e_1}$ [if you know]$_{e_2}$ [what I mean.]$_{e_3}$", where 'S' denotes Satellite and 'N' denotes Nucleus.

The globally optimized methods, on the other hand, learn scoring functions for subtrees and perform search over all possible trees to find the most probable tree for a text. Recent graph-based methods use neural models as scoring functions (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). Despite being more accurate than greedy parsers, these methods are generally slow having a polynomial time complexity ($O(n^3)$ or higher).

Recently, transition-based **top-down** parsing with Pointer Networks (Vinyals et al., 2015) has attained state-of-the-art results in both dependency and discourse parsing tasks with the same computational efficiency (Ma et al., 2018; Lin et al., 2019); thanks to the **encoder-decoder** architecture that makes it possible to capture information from the whole text and the previously derived subtrees,

[*] Linlin Liu is under the Joint PhD Program between Alibaba and Nanyang Technological University.
[†] Equal contribution.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1007–1017,
Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

while limiting the number of parsing steps to linear. However, the decoder of these parsers has a sequential structure, which may not yield the most appropriate **inductive bias** for deriving a hierarchical structure. For example, when decoding *"pens"* in Figure 1 in a top-down depth-first manner, the decoder state is directly conditioned on *"and"* as opposed to the states representing its parent *"sell"*. This on one hand may induce irrelevant information in the current state, on the other, as the text length gets longer, the decoder state at later steps tends to forget more relevant information due to long distance. Having an explicit hierarchical inductive bias should allow the model to receive more relevant information and help with the long-term dependency problem by providing shortcuts for gradient back-propagation.

In this paper, we propose a **Hierarchical Pointer Network (H-PtrNet)** parser to address the above mentioned limitations. In addition to the sequential dependencies, our parser also directly models the parent-child and sibling relationships in the decoding process. We apply our proposed method to both dependency and discourse parsing tasks. To verify the effectiveness of our approach, we conduct extensive experiments and analysis on both tasks. Our results demonstrate that in dependency parsing, our model outperforms in most of the languages. In discourse parsing, we push forward the state-of-the-art in all evaluation metrics. Furthermore, our results on the hardest task of relation labeling have touched human agreement scores on this task. We have released our code at https://ntunlpsg.github.io/project/parser/ptrnet-depparser/ for research purposes.

2 Background

2.1 Dependency Parsing

Dependency parsing is the task of predicting the existence and type of linguistic dependency relations between words in a sentence (Figure 1a). Given an input sentence, the output is a tree that shows relationships (*e.g.,* NOMINAL SUBJECT (NSUBJ), DETERMINER (DET)) between *head* words and words that modify those heads, called *dependents* or *modifiers*.

Approaches to dependency parsing can be divided into two main categories: greedy transition-based parsing and graph-based parsing. In both paradigms, neural models have proven to be more effective than feature-based models where selecting the composition of features is a major challenge. Kiperwasser and Goldberg (2016) proposed graph-based and transition-based dependency parsers with a Bi-LSTM feature representation. Since then much work has been done to improve these two parsers. Dozat and Manning (2017) proposed a **bi-affine** classifier for the prediction of arcs and labels based on graph-based model, and achieved state-of-the-art performance. Nguyen and Verspoor (2018) adopted a joint modeling approach by adding a Bi-LSTM POS tagger to generate POS tags for the graph-based dependency parser. Though transition-based methods are superior in terms of time complexity, they fail in capturing the global dependency information when making decisions. To address this issue, Andor et al. (2016) proposed a globally optimized transition-based model. Recently, by incorporating a stack within a pointer network, Ma et al. (2018) proposed a transition-based model and achieved state-of-the-art performance across many languages .

2.2 Discourse Parsing

Rhetorical Structure Theory or RST (Mann and Thompson, 1988) is one of the most influential theories of discourse, which posits a tree structure (called discourse tree) to represent a text (Fig. 1b). The leaves of a discourse tree represent contiguous text spans called Elementary Discourse Units (EDUs). The adjacent EDUs and larger units are recursively connected by coherence relations (*e.g.,* CONDITION, ATTRIBUTION). Furthermore, the discourse units connected by a relation are distinguished based on their relative importance — NUCLEUS refers to the core part(s) while SATELLITE refers to the peripheral one. Coherence analysis in RST consists of two subtasks: (*a*) identifying the EDUs in a text, referred to as **Discourse Segmentation**, and (*b*) building a discourse tree by linking the EDUs hierarchically, referred to as **Discourse Parsing**. This work focuses on the more challenging task of discourse parsing assuming that EDUs have already been identified. In fact, state-of-the-art segmenter (Lin et al., 2019) has already achieved 95.6 F_1 on RST discourse treebank, where the human agreement is 98.3 F_1.

Earlier methods have mostly utilized handcrafted lexical and syntactic features (Soricut and Marcu, 2003; Feng and Hirst, 2014; Joty et al., 2015; Wang et al., 2017). Recent approaches have

shown competitive results with neural models that are able to automatically learn the feature representations in an end-to-end fashion (Ji and Eisenstein, 2014; Li et al., 2014). Very recently, Lin et al. (2019) propose a parser based on pointer networks and achieve state-of-the-art performance.

Remark. Although related, the dependency and RST tree structures (hence the parsing tasks) are different. RST structure is similar to constituency structure. Therefore, the differences between constituency and dependency structures also hold here.[1] First, while dependency relations can only be between words, discourse relations can be between elementary units, between larger units or both. Second, in dependency parsing, any two words can be linked, whereas RST allows connections only between two adjacent units. Third, in dependency parsing, a head word can have multiple modifier words, whereas in discourse parsing, a discourse unit can be associated with only one connection. The parsing algorithm needs to be adapted to account for these differences.

2.3 Pointer Networks.

Pointer networks (Vinyals et al., 2015) are a class of encoder-decoder models that can tackle problems where the output vocabulary depends on the input sequence. They use attentions as pointers to the input elements. An encoder network first converts the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. At each time step t, the decoder takes the input from previous step, generates a decoder state $\boldsymbol{d}_t$, and uses it to attend over the input elements. The attention gives a $\mathrm{softmax}$ distribution over the input elements.

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i); \quad \boldsymbol{a}_t = \mathrm{softmax}(\boldsymbol{s}_t) \quad (1)$$

where $\sigma(.,.)$ is a scoring function for attention, which can be a neural network or an explicit formula like dot product. The model uses $\boldsymbol{a}_t$ to infer the output: $\hat{\mathrm{y}}_t = \arg\max(\boldsymbol{a}_t) = \arg\max p(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}, \theta)$ where θ is the set of parameters. To condition on y_{t-1}, the corresponding input $\boldsymbol{x}_{\mathrm{y}_{t-1}}$ is copied as the input to the decoder.

[1] There are also studies that use dependency structure to directly represent the relations between the EDUs; see (Muller et al., 2012; Li et al., 2014; Morey et al., 2018).

3 Hierarchical Pointer Networks

Before presenting our proposed hierarchical pointer networks, we first revisit how pointer networks have been used for parsing tasks.

3.1 Pointer Networks for Parsing.

Ma et al. (2018) and Lin et al. (2019) both use a pointer network as the backbone of their parsing models and achieve state-of-the-art performance in dependency and discourse parsing tasks, respectively. As shown in Figures 2(a) and 3, in both cases, the parsing algorithm is implemented in a top-down depth-first order. They share the same encoder-decoder structure. A bi-directional Recurrent Neural Network (RNN) encodes a sequence of word embeddings $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. The decoder implements a unidirectional RNN to greedily generate the corresponding tree. It maintains a stack to keep track of the units that yet need to be parsed, *i.e.,* head words for dependency parsing and larger units for discourse parsing. At each step t, the decoder takes out an element from the stack and generates a decoder state $\boldsymbol{d}_t$, which is in turn used in the pointer layer to compute the attention over the relevant input elements. In the case of dependency parsing, the representation of the head word is used to find its dependent. For discourse parsing, it uses the representation of the span to identify the break position that splits the text span into two subspans.

In addition to the tree structure, the parser also deals with the corresponding labelling tasks. Whenever the pointer network yields a newly created pair (*i.e.,* head-dependent in dependency parsing, two sub-spans in discourse parsing), a separate classifier is applied to predict the corresponding relation between them.

3.2 Limitations of Existing Methods

One crucial limitation of the existing models is that the decoder has a linear structure, although the task is to construct a hierarchical structure. This can be noticed in the Figures 2 and 3, where the current decoder state $\boldsymbol{d}_t$ is conditioned on the previous state $\boldsymbol{d}_{t-1}$ (see horizontal blue lines), but not on its parent's decoder state or siblings' decoder state, when it was pointed from its head. This can induce irrelevant information if the previous decoding state corresponds to an element that is not

relevant to the current element. For example, in Figure 2, the decoder state for pointing to "pens" is conditioned on the state used for pointing to "and", but not the one used for pointing to "sell", which is more relevant according to the dependency structure. Also, the decoder state for "sell" is far apart from the one for "pens". Therefore, more relevant information could be diminished in a sequential decoder, especially for long range dependencies.

3.3 Hierarchical Decoder

To address the above issues, we propose hierarchical pointer network (H-PtrNet), which poses a hierarchical decoder that reflects the underlying tree structure. H-PtrNet has the same encoder-decoder architecture as the original pointer network except that each decoding state $\boldsymbol{d}_t$ is conditioned directly on its parent's decoder state $\boldsymbol{d}_{p(t)}$ and its immediate sibling's decoder state $\boldsymbol{d}_{s(t)}$ in addition to the previous decoder state $\boldsymbol{d}_{t-1}$ and parent's encoder state $\boldsymbol{h}_{p(t)}$ (from input). Formally, the pointing mechanism in H-PtrNet can be defined as:

$$\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)}) \quad (2)$$

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i) \quad (3)$$

$$p(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}, \theta) = \mathrm{softmax}(\boldsymbol{s}_t) = \frac{\exp(s_{t,i})}{\sum_i \exp(s_{t,i})} \quad (4)$$

where $f(.)$ is a fusion function to combine the four components into a decoder state, and other terms are similarly defined as before for Eq. 1. Figure 2(b) shows an example of H-PtrNet decoder connections for dependency parsing.

The fusion function $f(.)$ can be implemented in multiple ways and may depend on the specific parsing task. More variants of the fusion function will be discussed in Section 4.

Decoder Time Complexity. Given a sentence of length n, the number of decoding steps to build a parse tree is linear. The attention mechanism at each decoding step computes an attention vector of length n. The overall decoding complexity is $O(n^2)$, which is same as the StackPointer Parser (Ma et al., 2018).

Remark. If we look at the decoding steps of the StackPointer Parser (Ma et al., 2018) more closely, we notice that it also takes the decoder state of the immediate sibling (when it points to itself). This decoder state represents the state after all its children are generated. Thus it contains information about its children. In contrast, in our model we consider the decoder state when the sibling was first generated from its parent. Therefore this state contains the sibling's parent information, which helps with capturing long term dependencies.

3.4 Model Specifics for Dependency Parsing

Figure 2(a) shows the encoding and decoding steps of H-PtrNet for dependency parsing. We use the same encoder as Ma et al. (2018) (red color).[2] Given a sentence, a convolutional neural network (CNN) is used to encode character-level representation of each word, which is then concatenated with word embedding and POS embedding vectors to generate the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Then a three-layer bi-directional LSTM encodes $\boldsymbol{X}$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. The decoder (blue color) is a single layer uni-directional LSTM, and also maintains a stack to track of the decoding status. At each decoding step t, the decoder receives the *encoder state* of the parent from the stack. In addition, it gets *decoder states* from three different sources: previous decoding step $\boldsymbol{d}_{t-1}$, parent $\boldsymbol{d}_{p(t)}$ and immediate sibling $\boldsymbol{d}_{s(t)}$.

Instead of simply feeding these three components to the decoder, we incorporate a gating mechanism to generalize the ability of our model to extract the most useful information. Eventually, the **fusion function** $f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)})$ in Eq. 2 is defined with a *gating mechanism*. We experimented with two different gating functions:

$$\boldsymbol{g}_t = \mathrm{sigmoid}(\boldsymbol{W}_{gd}\boldsymbol{d}_{t-1} + \boldsymbol{W}_{gp}\boldsymbol{d}_{p(t)} + \boldsymbol{W}_{gs}\boldsymbol{d}_{s(t)} + \boldsymbol{b}_g) \quad (5)$$

$$\boldsymbol{g}_t = \mathrm{sigmoid}(\boldsymbol{W}_{gp}(\boldsymbol{d}_{t-1} \odot \boldsymbol{d}_{p(t)}) + \boldsymbol{W}_{gs}(\boldsymbol{d}_{t-1} \odot \boldsymbol{d}_{s(t)}) + \boldsymbol{b}_g) \quad (6)$$

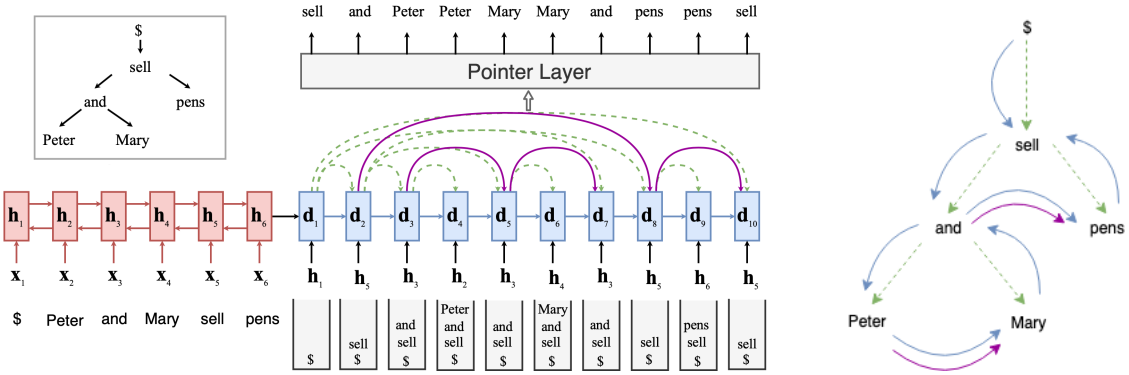
where $\boldsymbol{W}_{gp}$, $\boldsymbol{W}_{gs}$, $\boldsymbol{W}_{gd}$ and $\boldsymbol{b}_g$ are the gating weights. The fusion function is then defined as

$$\boldsymbol{h}'_t = \tanh(\boldsymbol{W}_d\boldsymbol{d}_{t-1} + \boldsymbol{W}_p\boldsymbol{d}_{p(t)} + \boldsymbol{W}_s\boldsymbol{d}_{s(t)}) \quad (7)$$

$$\boldsymbol{h}''_t = \boldsymbol{g}_t \odot \boldsymbol{h}'_t \quad (8)$$

$$\boldsymbol{d}_t = \mathrm{LSTM}(\boldsymbol{h}''_t, \boldsymbol{h}_{p(t)}) \quad (9)$$

[2] https://github.com/XuezheMax/NeuroNLP2



(a) Model Structure (b) Decoder Connections

Figure 2: (a) H-PtrNet for dependency parsing. To reduce visual clutter, we do not show the attention scores over all the input elements at each pointing step, rather show only the pointed element. Figure (b) shows the decoder connections in our H-PtrNet model. The StackPointer network of Ma et al. (2018) has a sequential decoder (shown as blue straight lines). The Green dash lines indicate parent connections and purple solid lines denote the immediate sibling connections in our model.

where $\boldsymbol{W}_d$, $\boldsymbol{W}_p$, $\boldsymbol{W}_s$ are the weights to get the intermediate hidden state $\boldsymbol{h}'_t$, and $\boldsymbol{g}_t$ is a gate to control the information flow from the three decoder states LSTM is the LSTM layer that accepts $\boldsymbol{h}''_t$ as the the hidden state and $\boldsymbol{h}_{p(t)}$ as its input. The LSTM decoder state $\boldsymbol{d}_t$ is then used to compute the attention distribution over the encoder states in pointer layer.

Pointer and Classifier. Same as Ma et al. (2018), the pointer and the label classifier are implemented as **bi-affine** layers. Formally, the scoring function $\sigma(\boldsymbol{d}_t, \boldsymbol{h}_i)$ in Eq. 3 is defined as:

$$s_{t,i} = g_1(\boldsymbol{d}_t)^T \boldsymbol{W} g_2(\boldsymbol{h}_i) + \boldsymbol{U} g_1(\boldsymbol{d}_t) + \boldsymbol{V} g_2(\boldsymbol{h}_i) + \boldsymbol{b} \quad (10)$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the weights, and $g_1(.)$ and $g_2(.)$ are two single layer MLPs with ELU activations. The dependency label classifier has the same structure as the pointer. More specifically,

$$p(\mathrm{y}_l|\boldsymbol{X}) = \mathrm{softmax}(g'_1(\boldsymbol{d}_t)^T \boldsymbol{W}_c g'_2(\boldsymbol{h}_k) + \boldsymbol{U}_c g'_1(\boldsymbol{d}_t) + \boldsymbol{V}_c g'_2(\boldsymbol{h}_k) + \boldsymbol{b}_c) \quad (11)$$

where $\boldsymbol{h}_k$ is the encoder state of the dependent word, $\boldsymbol{d}_t$ is the decoder state of the head word, $\boldsymbol{W}_c$, $\boldsymbol{U}_c$ and $\boldsymbol{V}_c$ are the weights, and $g'_1(.)$ and $g'_2(.)$ are two single layer MLPs with ELU activations.

Partial Tree Information. Similar to Ma et al. (2018), we provide the decoder at each step with higher order information about the parent and the sibling of the current node.

3.5 Model Specifics for Discourse Parsing.

For discourse parsing, our model uses the same structure as Lin et al. (2019).[3] The encoder is a 5-layer bidirectional RNN based on Gated Recurrent Units (BiGRU) (Cho et al., 2014). As shown in Figure 3, after obtaining a sequence of encoder hidden states representing the words, the last hidden states of the EDUs (*e.g.*, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$, $\boldsymbol{h}_5$, $\boldsymbol{h}_6$, $\boldsymbol{h}_8$ and $\boldsymbol{h}_{10}$) are taken as the EDU representations, generating a sequence of EDU representations $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m)$ for the input sentence.

Our hierarchical decoder is based on a 5-layer unidirectional GRU. The decoder maintains a **stack** to keep track of the spans that need to be parsed further. At time step t, the decoder takes the text span (*e.g.*, $\boldsymbol{e}_{i:j}$) representation from the top of the stack and receives the corresponding parent decoder state $\boldsymbol{d}_{p(t)}$, sibling decoder state $\boldsymbol{d}_{s(t)}$ and previous decoder state $\boldsymbol{d}_{t-1}$ as the input to generate a current decoder state $\boldsymbol{d}_t$. For discourse parsing, we apply Eq. 7 and 9 (with GRU) to implement the fusion function $f(.)$ and to get the decoder state $\boldsymbol{d}_t$.[4] The decoder state is then used in the pointer layer to compute the attention score over the current text span (*e.g.*, $\boldsymbol{e}_{i:j}$) in order to find the position k to generate a new split $(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$. The parser then applies a relation classifier $\Phi(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$ to predict the relation and the nuclearity labels for the the new split.

[3] https://ntunlpsg.github.io/project/parser/pointer-net-parser

[4] Adding gating mechanism did not give any gain, rather increased the number of parameters.

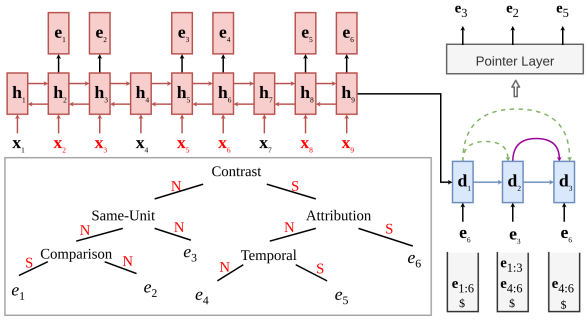


Figure 3: H-PtrNet for discourse parsing. The input symbols in red ($\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_8, \mathbf{x}_9$) represent last words of the respective EDUs. To avoid visual clutter, we do not show the attention distributions over the EDUs, rather only show the decisions. Green dash lines indicate parent connections and purple solid lines denote the immediate sibling connections.

For pointing, the parser uses a simple **dot product** attention. For labeling, it uses a **bi-affine classifier** similar to the one in Eq. 11. It takes the representations of two spans (*i.e.,* $\boldsymbol{e}_k$ for $\boldsymbol{e}_{i:k}$, $\boldsymbol{e}_j$ for $\boldsymbol{e}_{k+1:j}$) as input and predicts the corresponding relation between them. Whenever the length of any of the newly created span ($\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_{k+1:j}$) is larger than two, the parser pushes it onto the stack for further processing. Similar to dependency parsing, the decoder is also provided with **partial tree information** – the representations of the **parent** and the immediate **sibling**.

3.6 Objective Function

Same as Ma et al. (2018) and Lin et al. (2019), our parsers are trained to minimize the total log loss (cross entropy) for building the right tree **structure** for a given sentence $\boldsymbol{X}$. The structure loss $\mathcal{L}$ is the pointing loss for the pointer network:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log P_\theta(\mathrm{y}_t | \mathrm{y}_{<t}, \boldsymbol{X}) \qquad (12)$$

where θ denotes the model parameters, $y_{<t}$ represents the subtrees that have been generated by our parser at previous steps, and T is the number needed for parsing the whole sentence (*i.e.,* number of words in dependency parsing and spans containing more than two EDUs in discourse parsing). The label classifiers are trained simultaneously, so the final loss function is the sum of structure loss (Eq. 12) and the loss for label classifier.

4 Experiments

In this section, we describe the experimental details about dependency parsing and discourse parsing, as well as the analysis on both tasks.

Apart from the two gating-based fusion functions described in Section 3 (Eq. 5-6), we experimented with three different versions of our model depending on which connections are considered in the decoder. We append suffixes **P** for *parent*, **S** for *sibling* and **T** for *temporal* to the model name (H-PtrNet) to denote different versions.

- **H-PtrNet-P**: The H-PtrNet model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{h}_{p(t)})$, where the decoder receives hidden (decoder) state only from the parent ($\boldsymbol{d}_{p(t)}$) in each decoding step. Note that $\boldsymbol{h}_{p(t)}$ is the encoder state of the parent.

- **H-PtrNet-PS**: The H-PtrNet model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{h}_{p(t)})$, where the decoder receives the hidden states from both the parent and sibling in each decoding step.

- **H-PtrNet-PST**: This is the full model with fusion function $\boldsymbol{d}_t = f(\boldsymbol{d}_{p(t)}, \boldsymbol{d}_{s(t)}, \boldsymbol{d}_{t-1}, \boldsymbol{h}_{p(t)})$ (Eq. 2). In this model, the decoder receives the hidden states from its parent, sibling and previous step in each decoding step.

4.1 Dependency Parsing

Dataset. We evaluate our model on the English Penn Treebank (PTB v3.0) (Marcus et al., 1994), which is converted to Stanford Dependencies format with Stanford Dependency Converter 3.3.0 (Schuster and Manning, 2016). To make a thorough empirical comparison with previous studies, we also evaluate our system on seven (7) languages from the Universal Dependency (UD) Treebanks[5] (version 2.3).

Metrics. We evaluate the performance of our models with unlabeled attachment score (UAS) and labeled attachment score (LAS). We ignore punctuations in the evaluation for English.

Experimental Settings. We use the same setup as Ma et al. (2018) in the experiments for English Penn Treebank and UD Treebanks. For a fair comparison, we rerun their model with the hyperparameters provided by the authors on the same machine as our experiments. For all the languages,

[5] http://universaldependencies.org/

	StackPtr (code)		H-PtrNet-PST (Gate)		H-PtrNet-PST (SGate)	
	UAS	LAS	UAS	LAS	UAS	LAS
bg	94.17±0.11	90.63±0.06	94.20±0.16	90.70±0.14	**94.50±0.16**	**91.01±0.20**
ca	**93.82±0.06**	**91.99±0.07**	93.78±0.03	91.92±0.03	93.67±0.06	91.82±0.07
en	90.97±0.07	89.06±0.08	**91.03±0.19**	**89.07±0.16**	90.94±0.12	89.01±0.12
de	87.97±0.20	83.75±0.21	**88.14±0.22**	**83.89±0.26**	88.06±0.17	83.83±0.13
fr	91.57±0.23	88.76±0.21	91.63±0.15	88.70±0.14	**91.69±0.07**	**88.80±0.11**
it	93.76±0.13	92.00±0.08	93.73±0.08	91.90±0.10	**93.88±0.05**	**92.09±0.02**
ro	91.15±0.12	85.54±0.13	**91.34±0.18**	**85.73±0.22**	91.09±0.09	85.36±0.10

Table 1: Dependency parsing results on 7 UD Treebanks. **StackPtr (code)** denotes the experiments we rerun on our machine. **H-PtrNet-PST (Gate)** and **H-PtrNet-PST (SGate)** are H-PtrNet models with gating mechanism.

we follow the standard split for training, validation and testing. It should be noted that Ma et al. (2018) used UD Treebanks 2.1, which is not the most up-to-date version. Therefore, during experiments, we rerun their codes with UD Treebanks 2.3 to match our experiments. To be specific, we use structured-skipgram (Ling et al., 2015) for English and German, while Polyglot embedding (Al-Rfou et al., 2013) for the other languages. Adam optimizer (Kingma and Ba, 2015) is used as the optimization algorithm. We apply 0.33 dropout rate between layers of encoder and to word embeddings as well as Eq. 8. We use beam size of 10 for English Penn Treebank, and beam size of 1 for UD Treebanks. The gold-standard POS tags is used for English Penn Treebank. We also use the universal POS tags (Petrov et al., 2011) provided in the dataset for UD Treebanks. See Appendix for a complete list of hyperparameters.

Results on UD Treebanks. We evaluate on 7 different languages from the UD Treebanks: 4 major ones: English (en), German (de), French (fr), and Italian (it), and 3 relatively minor ones: Bulgarian (bg), Catalan (ca), and Romanian (ro). Table 1 shows the results. We refer to the results of our run of the code released by Ma et al. (2018) as StackPtr (code).[6] StackPtr (code) and our models are trained in identical settings making them comparable. H-PtrNet-PST (Gate) (Eq. 5) and H-PtrNet-PST (SGate) (Eq. 6) are H-PtrNet models with gating mechanism. Element wise product in Eq. 6 has the effect of similarity comparison, so we denote it as **SGATE**. With gating mechanism, our model shows consistent improvements against the baseline on bg, en, de, fr, it and ro. We also tested H-PtrNet-PS on these 7 languages, but the performances are worse than StackPtr.

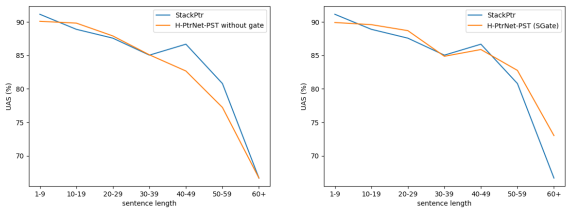
[6] We do not directly report the results from their paper because we use a different version of the UD Treebanks.

Approach	UAS	LAS
Baselines		
StackPtr (paper)	96.12±0.03	95.06±0.05
StackPtr (code)	95.94±0.03	94.91±0.05
Proposed Model		
H-PtrNet-PST (Gate)	96.03±0.02	94.99±0.02
H-PtrNet-PST (SGate)	96.04±0.05	95.00±0.06
H-PtrNet-PS (Gate)	**96.09±0.05**	**95.03±0.03**

Table 2: Dependency parsing results on English Penn Treebank v3.0.

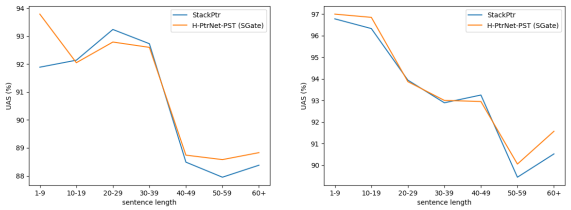
Results on English Penn Treebank. Table 2 presents the results on English Penn Treebank. StackPtr (paper) refer to the results reported by Ma et al. (2018), and StackPtr (code) is our run of their code in identical settings as ours. Our model H-PtrNet-PST (Gate) outperforms the baseline by 0.09 and 0.08 in terms of UAS and LAS, respectively. Performance of H-PtrNet-PST (SGate) is close to that of H-PtrNet-PST (Gate), though we see slight improvement. We also test H-PtrNet-PS (Gate), the model with parent and sibling connections only, which further improves the performance to 96.09 and 95.03 in UAS and LAS.

Performance Analysis. To make a thorough analysis of our model, we breakdown UAS in terms of sentence lengths to compare the performance of our model and StackPtr. We first take the performance on UD German as an example,



(a) H-PtrNet-PST (no gating) (b) H-PtrNet-PST (SGate)

Figure 4: UD German parsing performance in terms of sentence length.



(a) UD French (b) UD Italian

Figure 5: Performance analysis on French and Italian based on H-PtrNet-PST (SGate).

which is shown in Figure 4. The blue line shows the performance of StackPtr, and the orange line shows the performance of our model. From Figure 4(a) we can see that our model without gate performs better on relatively short sentences (10 to 29 words), however, the accuracy drops on longer sentences. The reason could be that adding parent and sibling hidden states to decoder may amplify error accumulation from early parsing mistakes.

Figure 4(b) shows the performance of our model with SGATE (Eq. 6), where we can see that the performance on long sentences has been improved significantly. In the meanwhile, it still maintains higher accuracy than StackPtr on the short sentences (10 to 29 words). Figure 5 shows two more examples, again, from which we can see that our model with SGATE tends to outperform StackPtr on longer sentences.

4.2 Discourse Parsing

Dataset. We use the standard RST Discourse Treebank (RST-DT) (Carlson et al., 2002), which contains discourse annotations for 385 news articles from Penn Treebank (Marcus et al., 1994). We evaluate our model in sentence-level parsing, for which we extract all the well-formed sentence-level discourse trees from document-level trees.

Approach	Span	Nuclearity	Relation
Human Agreement	95.7	90.4	83.0
Baselines			
Joty et al. (2012)	94.6	86.9	77.1
Ji and Eisenstein (2014)	93.5	81.3	70.5
Wang et al. (2017)	95.6	87.8	77.6
Pointer Net† (Lin et al., 2019)	$97.39_{\pm 0.1}$	$91.01_{\pm 0.4}$	$81.08_{\pm 0.4}$
Pointer Net§ (Lin et al., 2019)	$97.14_{\pm 0.1}$	$91.00_{\pm 0.3}$	$81.29_{\pm 0.2}$
Proposed Model			
H-PtrNet-P†	$\mathbf{97.68}_{\pm 0.03}$	$91.86_{\pm 0.2}$	$81.82_{\pm 0.2}$
H-PtrNet-P§	$97.51_{\pm 0.08}$	$91.98_{\pm 0.1}$	$82.11_{\pm 0.2}$
H-PtrNet-PS†	$97.56_{\pm 0.1}$	$91.52_{\pm 0.3}$	$82.05_{\pm 0.5}$
H-PtrNet-PS§	$97.35_{\pm 0.1}$	$91.78_{\pm 0.1}$	$82.35_{\pm 0.2}$
H-PtrNet-PST†	$97.56_{\pm 0.06}$	$91.97_{\pm 0.3}$	$82.37_{\pm 0.4}$
H-PtrNet-PST§	$97.48_{\pm 0.2}$	$\mathbf{92.01}_{\pm 0.2}$	$\mathbf{82.77}_{\pm 0.2}$

Table 3: Discourse parsing results with gold segmentation. † denotes the models selected based on Span. § denotes the models selected based on Relation. We run all the experiments for three times and report the averages and the standard deviations.

In all, the training data contains 7321 sentences, and the testing data contains 951 sentences. These numbers match the statistics reported by Lin et al. (2019). We follow the same settings as in their experiments and randomly choose 10% of the training data for hyperparameter tuning.

Metric and Relation Labels. Following the standard in RST parsing, we use the unlabeled (Span) and labeled (Nuclearity, Relation) metrics proposed by Marcu (2000). We only present F_1-score for space limitations. Following the previous work, we attach the nuclearity labels (NS, SN, NN) to 18 discourse relations, together giving 39 distinctive relation labels.

Experimental Settings. Since our goal is to evaluate our parsing method, we conduct the experiments based on gold EDU segmentations. We compare our results with the recently proposed pointer network based parser of Lin et al. (2019) (Pointer Net). However, unlike their paper, we report results for both cases: (i) when the model was selected based on the best performance on Span identification; and (ii) when it was selected based on the relation labeling performance on the development set. We retrain their model for both settings. We also apply Adam optimizer as optimization algorithm and ELMo (Peters et al., 2018) with 0.5 dropout rate as word embeddings.

Results. We present the results in Table 3. In discourse parsing, the number of EDUs in a sentence is relatively small compared to the sentence lengths (in words) in dependency parsing. Based on the observation in dependency parsing that the performance of H-PtrNet may drop for longer sentences due to parent error accumulation, we expect that in discourse parsing, this should not be the case since the the number of parsing steps is much smaller compared to that of dependency parsing.

We first consider the models that were selected based on Span performance (models with † superscript). H-PtrNet-P, with only parent connection, outperforms the baseline in all three tasks. It achieves an absolute improvement of 0.29 F_1 in span identification compared to the baseline. Considering the performance has already exceeded the human agreement of 95.7 F_1, this gain is remarkable. Thanks to the higher accuracy on finding the right spans, we also achieve 0.85 and 0.74 absolute improvements in Nuclearity and Relation tasks, respectively. By adding the sibing and temporal connections, we test the performance of our full model, H-PtrNet-PST. The performance on Span is 0.17 F_1 higher than the baseline. However, it is not on par with our H-PtrNet-P. But, it is not surprising since we adopt binary tree structures in discourse parsing, which means the sibling information could be redundant in most cases. This also accords with our previous assumption that parent connections may bring enough information to decode RST trees.

Now we consider the models that were selected based on Relation labeling performance (models with § superscript). We achieve significant improvement in Relation compared to the baseline. Eventually the parser yields an F_1 of 82.77, which is very close to the human agreement (83.0 F_1). We observe that the performance in H-PtrNet-PS and H-PtrNet-PST is better than the H-PtrNet-P. As the relation classifier and the pointer network share the same encoder information (Sec. 3), we believe that richer decoder information leads the model to learn better representations of the text spans (encoder states) and further leads to a better performance in relation labeling.

We further analyze the performance of our proposed model in terms of number of EDUs. We present the F_1 score in Span of H-PtrNet-P and H-PtrNet-PST as well as the baseline in Figure 6. It can be observed that both H-PtrNet-P and H-PtrNet-PST outperform the baseline with respect to almost every number of EDUs. Moreover, we can see that the H-PtrNet-P performs better in most of the cases, which once again conforms to our assumption that parent information is enough to decode RST trees. However, as discussed in our dependency parsing experiments, when the number of words (EDUs for discourse parsing) increases, the model may suffer from error accumulation from early parsing. Hence, H-PtrNet-PST tends to perform better when EDU number becomes large.

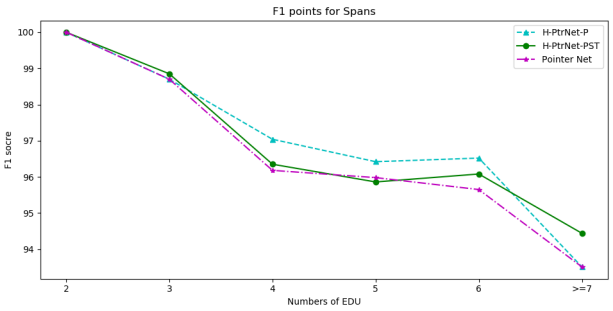


Figure 6: F_1 score in Span for different EDU numbers.

5 Conclusions

In this paper, we propose hierarchical pointer network parsers and apply them to dependency and discourse parsing tasks. Our parsers address the limitation of previous methods, where the decoder has a sequential structure while it is decoding a hierarchical tree structure, by allowing more flexible information flow to help the decoder receive the most relevant information. For both tasks, our parsers outperform existing methods and set new state-of-the-arts of the two tasks. The broken-down analysis clearly illustrates that our parsers perform better for long sequences, complying with the motivation of our model.

Acknowledgements

This research is partly supported by the Alibaba-NTU Singapore Joint Research Institute, Nanyang Technological University (NTU), Singapore. Shafiq Joty would like to thank the funding support from his Start-up Grant (M4082038.020).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word represen-

tations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank (RST–DT) LDC2002T07. *Linguistic Data Consortium, Philadelphia*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 340–345, Copenhagen, Denmark. ACL.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. ACL.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41:3:385–435.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35. ACL.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A Unified Linear-Time Framework for Sentence-Level Discourse Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages xx–xx, Florence, Italy. ACL.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414. Association for Computational Linguistics.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of SIGIR*, pages 137–144.

Daniel Marcu. 2000. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on RST discourse parsing and evaluation. *American Journal of Computational Linguistics*, 44(2):197–235.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai, India. The COLING 2012 Organizing Committee.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint POS tagging and dependency parsing. *CoRR*, abs/1807.03955.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL'03, pages 149–156, Edmonton, Canada. ACL.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.