IRE^{2}F: Rethinking Effective Refinement in Language Structure Prediction via Efficient Iterative Retrospecting and Reasoning

Zuchao Li[1,2], **Xingyi Guo**[1], **Letian Peng**[3], **Lefei Zhang**[1,2], **and Hai Zhao**[4,*]
[1]National Engineering Research Center for Multimedia Software,
School of Computer Science, Wuhan University, Wuhan, 430072, P. R. China
[2]Hubei Luojia Laboratory, Wuhan 430072, P. R. China
[3]Computer Science and Engineering, University of California San Diego
[4]Department of Computer Science and Engineering, Shanghai Jiao Tong University
{zcli-charlie, guoxingyi, zhanglefei}@whu.edu.cn, lepeng@ucsd.edu, zhaohai@cs.sjtu.edu.cn

Abstract

Refinement plays a critical role in language structure prediction, a process that deals with complex situations such as structural edge interdependencies. Since language structure prediction is usually modeled as graph parsing, typical refinement methods involve taking an initial parsing graph as input and refining it using language input and other relevant information. Intuitively, a refinement component, i.e., refiner, should be lightweight and efficient, as it is only responsible for correcting faults in the initial graph. However, current refiners significantly burden the parsing process due to their reliance on the time-consuming encoding-decoding procedure on the language input and graph. To make the refiner more practical for real-world applications, this paper proposes a lightweight but effective iterative refinement framework, IRE^{2}F, based on the iterative retrospecting and reasoning without involving the re-encoding process on the graph. IRE^{2}F iteratively refine the parsing graph based on interaction between graph and sequence and efficiently learns the shortcut to update the sequence and graph representations in each iteration. The shortcut is calculated based on the graph representation in the latest iteration. IRE^{2}F reduces the number of refinement parameters by 90% compared to the baseline iterative refiners. Experiments on various language structure prediction tasks show that IRE^{2}F performs comparably or better than current state-of-the-art refiners, with a significant increase in efficiency.

1 Introduction

Language structure prediction is an important area of research in natural language processing (NLP), which involves using computational techniques to analyze and understand the complex structure of natural languages [Sun *et al.*, 2021]. It is usually modeled as graph parsing, which involves analyzing the grammatical graph of a sentence or text, or identifying the relationships between words and phrases in a document.

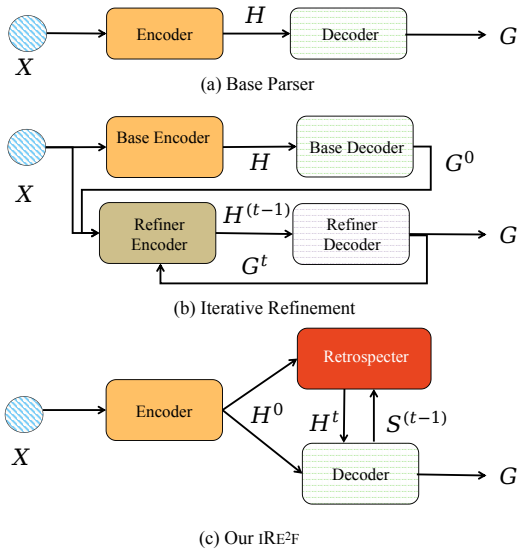

Figure 1: Architectures of base parser (a), parser with iterative refinement (b) [Lyu *et al.*, 2019a; Mohammadshahi and Henderson, 2021], and parser with our proposed IRE^{2}F framework. Arrows indicates the flows in each iteration.

Parsing involves analyzing and representing the grammatical structure of a sentence or text in a structured format like a tree or graph. A typical parser consists of an encoder, which vectorizes and contextualizes the input, and a decoder, which scores and generates the structured output. The initial parsing output, often referred to as the draft parse, may not be fully accurate or complete, as it is based on limited information and may contain errors or inconsistencies. Refinement is a process that aims to improve the accuracy and completeness of the parse by incorporating additional information and correcting any errors or inconsistencies. Refinement is important because it helps to improve the overall quality and reliability of the parse, which is critical for accurate language structure prediction.

Iterative refinement is a technique introduced by Lyu *et al.* that aims to capture interdependencies between dependents and corresponding labels in graph parsing tasks, which

is used to refine the parsing graph through fault correction. The proposed framework takes the initial sequence representation $\mathcal{H}^0$ from the base parser, as well as the sequence and graph representations $\mathcal{H}^{(t-1)}$ and $\mathcal{G}^{(t-1)}$ from the latest iteration, as input. The refiner then reasons a new parsing graph $\mathcal{G}^t$ based on this input:

$$\mathcal{G}^t = \textbf{REFINER}(\mathcal{H}^{(t-1)}, \mathcal{G}^{(t-1)}, \mathcal{H}^0).$$

The above refining process involves capturing missing interdependencies in the input draft parsing graph to reason better parsing graphs in each iteration. Mohammadshahi and Henderson further proposed using a graph-to-graph encoder to improve the performance of the refiner.

There is no doubt that iterative refiners can play a significant role in the performance improvement of graph parsing. However, the current implementation of refiners adds a significant computational burden to the parser, which goes against the original idea that refiners are only used to correct a small number of faults in the draft parses. Specifically, each iteration in current refiners involves repeating the encoding-decoding procedure in the base parser, with the only difference being the inclusion of a merging step between the representations from the initial output and previous iterations.

The inefficiency of current iterative refiners is due to the re-encoding procedure, which involves repeating the time-consuming encoding process (such as BiLSTM in [Lyu *et al.*, 2019a] or Graph Transformer in [Mohammadshahi and Henderson, 2021]) in each refining iteration. Motivated by the fact that errors in graph parsing is non-dominant, we propose an efficient refiner framework, IRE^{2}F, for graph parsing via iterative retrospecting and reasoning. As shown in Figure 1, our framework excludes the expensive encoder from the refiner. Instead of refining using $\mathcal{H}^{(t-1)}$ and $\mathcal{G}^{(t-1)}$ from the latest iteration, our framework directly learns a retrospecter with a shortcut $\mathcal{R}^t$ from $\mathcal{G}^{(t-1)}$ to perform retrospecting and reasoning.

We compare the proposed IRE^{2}F refiner with previous state-of-the-art iterative refiners and a high-order refiner on various language structure prediction tasks. These tasks include dependency parsing for syntax, semantic dependency parsing and semantic role labeling for semantics. Our experiments demonstrate that IRE^{2}F reduces the computational burden in each refinement iteration to 10% while achieving comparable or better performances. It also has a simpler structure with significantly fewer parameters, indicating that the improvement is due to the refinement mechanism rather than a large number of extra parameters. We also find that combining IRE^{2}F with higher-order refinement leads to further improvement, demonstrating the benefits of multi-stage refinement.

2 Related Work

2.1 Language Structure Prediction

Early work in language structure prediction focused on developing rule-based systems [Bocharova, 2008], which involved manually defining the rules for analyzing and representing the structure of a language. More recently, there has been a shift towards the use of machine learning techniques for language structure prediction, such as statistical parsing [Nivre *et al.*, 2009] and graph-based parsing [Chen and Manning, 2014; Li *et al.*, 2022b].

Neural dependency parsing has become the dominant approach for language structure prediction and can be divided into two main categories: transition-based parsers and graph-based parsers. Transition-based neural parsers use a word buffer and a word stack to monotonically decide the next action to take , while graph-based parsers assign global scores to edges and labels in order to directly construct the full dependency graph in one-shot. Early graph-based neural parsers such as [Kiperwasser and Goldberg, 2016] and [Hashimoto *et al.*, 2016] used BiLSTMs and MLPs to contextualize and compute the global scores on the possible pair representations. Since the introduction of the deep biaffine attention by [Dozat and Manning, 2017], dot product with bias mechanism have been widely used in dependency parsing. Similar to syntactic dependency parsing, graph-based parsers have also been successfully applied to semantic dependency parsing [Peng *et al.*, 2017; Dozat and Manning, 2018; Jia *et al.*, 2020; Kurita and Søgaard, 2019].

2.2 Refinement

Iterative refinement has been shown to be effective in language generation tasks such as machine translation [Lee *et al.*, 2020] and long text generation [Hua and Wang, 2020], which is resulted from helping understanding complex sequence [Li *et al.*, 2022a]. In natural language understanding tasks, iterative refinement is often used for label annotation. In the work of [Gui *et al.*, 2020], researchers used a variational structure to evaluate the uncertainty of labels annotated in the first run and leveraged this uncertainty information to guide the refinement runs to only correct labels with high uncertainty and avoid changing labels with low uncertainty.

In language structure prediction, there are two styles of iterative refinement for graph-based parsers. In [Lyu *et al.*, 2019a], intermediate representations are extracted and passed to subsequent iterations without being decoded into explicit graphs or trees. While in [Mohammadshahi and Henderson, 2021], graphs decoded from the base parser or refiner are combined with the sentence as new inputs and fed through a Transformer encoder to generate new representations. In this work, we adopt the former style of refinement and design a novel refiner framework via efficient retrospecting and reasoning on the intermediate representations.

3 IRE^{2}F Framework

Our IRE^{2}F framework, like the general refiner framework, consists of both a base parser and refinement components. The base parser is responsible for encoding of the input and initial parsing, while the refinement component focuses on correcting any errors or faults in the initial parsing or encoding and improving the overall structure of the output. The refinement component takes the output of the base parser, as well as additional input information, and uses it to iteratively refine the structure until it meets the desired criteria.

3.1 Base Parser

To create a base parser for the initial parsing, we follow the model structure of biaffine parser (BIAF) proposed in [Dozat and Manning, 2017]. Given an input sentence with n words X, we first embed the words and their linguistic features X_{feat}. We also obtain pre-trained contextualized representations from a pre-trained language model (PLM) to a strong baseline. Finally, we concatenate these representations to obtain the initial representation $\mathcal{H}^0$:

$$\mathcal{H}^0 = \textsc{Embed}(X) \oplus \textsc{Embed}(X_{feat}) \oplus \text{PLM}(X)$$

where $\oplus$ refers to the concatenation operation, and $\textsc{Embed}(\cdot)$ represents the randomly initialized embedding layer.

The BiLSTM has a long history of being a successful choice for encoding sequential data and has been widely used in various language structure prediction tasks, demonstrating strong performance. Therefore, we chose to use the BiLSTM as the encoder for our base model. In our work, we contextualize the initial representations $\mathcal{H}^0$ using a stacked multi-layer BiLSTM encoder to produce $\mathcal{C}$. We then project $\mathcal{C}$ using two linear layers to obtain head and dependent role-specific representations $\mathcal{C}_H$ and $\mathcal{C}_D$.

$$\begin{aligned}\mathcal{C} &= \text{BiLSTM}_{base}(\mathcal{H}^0),\\ \mathcal{C}_H &= \mathbb{W}^H_{base}\mathcal{C} + b^H_{base},\\ \mathcal{C}_D &= \mathbb{W}^D_{base}\mathcal{C} + b^D_{base},\end{aligned}$$

where $\mathbb{W}^H_{base}$ and $\mathbb{W}^D_{base}$ are learnable weight parameters, b^H_{base} and b^D_{base} are bias items.

In language structure prediction as graph parsing, relationships between words are modeled as scores of edges between graph nodes. Therefore, performing pair scoring on role-specific representations is fundamental to enable graph reasoning. Biaffine attention mechanism has demonstrated excellent pair scoring ability since it was proposed. Therefore, in the base parser, we fed head and dependent representations into a biaffine attention scorer to get the scores for the existence and label of each possible edge.

$$\begin{aligned}\text{BIAF}(\mathcal{C}_H, \mathcal{C}_D) = \ & \mathcal{C}_H^T \mathbb{W}_{\text{B}} \mathcal{C}_D + \mathbb{W}_{\text{H}} \mathcal{C}_H + \mathbb{W}_{\text{D}} D + b,\\ \mathcal{S} = \ & \text{BIAF}_{base}(\mathcal{C}_H, \mathcal{C}_D),\end{aligned}$$

where $\mathbb{W}_{\text{B}} \in \mathbb{R}^{d\times(c+1)\times d}$, $\mathbb{W}_{\text{H}}, \mathbb{W}_{\text{D}} \in \mathbb{R}^{(c+1)\times d}$, and b is a bias item. Here c refers to the number of relationship categories, d is the dimension of input representations. The $(c+1)$ is due to the introduction of a *null* relation, and $\mathcal{S}$ is recognized as the graph representation which can be further decoded to a graph $\mathcal{G}$.

3.2 Rethinking Refinement

The refinement methods in the literature can be divided into three categories: Sequence Representation-Driven, Joint Sequence-Structure Representation-Driven, and Higher-Order Representation-Driven.

Sequence Representation-Driven (SRD) focuses on tuning the graph representation generated by the base parser based on the iteratively-updated sequence representation to achieve the purpose of refinement. It involves using the output of the base model as input to the refinement process, and then updating the representation based on this draft parses and iterative encoding.

In [Lyu *et al.*, 2019a], the predicted logits from the previous iteration are merged with re-encoded representations in the current iteration. For example, in iteration t, this process involves using a BiLSTM encoder to re-contextualize the initial representation $\mathcal{H}^0$ to $\mathcal{C}^t$, which is then projected to $\mathcal{C}^t_H$ and $\mathcal{C}^t_D$ as in the base parser. The refiner merges the current representation of the parsing graph S^t in the iteration t with $\mathcal{C}^t_H$, $\mathcal{C}^t_D$ using specific merge operations (like multiple linear projections) into a tuned graph representation $\tilde{\mathcal{S}}^t \in \mathbb{R}^{n\times n\times d}$:

$$\tilde{\mathcal{S}}^t = \textsc{Merge}^t(S^{(t-1)}, \mathcal{C}^t_H, \mathcal{C}^t_D),$$

where $\textsc{Merge}(\cdot)$ is the process of combining the previous parsing output $\mathcal{S}^{(t-1)}$ with the current step's representataions $\mathcal{C}^t_H$ and $\mathcal{C}^t_D$ during the interative refinement process. In [Lyu *et al.*, 2019a], this is achieved by summing up the probability mass of all labels in $\mathcal{S}^{(t-1)}$ and fusing them through a linear layer projection, which is then added to $\mathcal{C}^t_H$ and $\mathcal{C}^t_D$. $\tilde{\mathcal{S}}^t$ is further mapped by a linear projection into $\mathcal{S}^t \in \mathbb{R}^{n\times n\times(c+1)}$.

$$\mathcal{S}^t = \mathbb{W}^{G,t}\tilde{\mathcal{S}}^t + b^{G,t}$$

where $\mathbb{W}^{G,t}$ and $b^{G,t}$ are learnable parameters.

Joint Sequence-Structure Representation-Driven (JSSRD) focuses on completely re-encoding the decoded draft parse and the input sequence to obtain a new joint representation to identify errors or faults in the graph for refinement. In the proposed method of [Mohammadshahi and Henderson, 2021], the draft graph from the base parser or last iteration is passed through a graph-informed Transformer encoder to produce updated representations, which is then used to produce the new parsing graph. This differs from the approach in [Lyu *et al.*, 2019a], which only considers the representations from the base parser and the previous iteration in the refinement process, rather than the draft graph itself.

$$\begin{aligned}\mathcal{Z}^t &= \textsc{GraphTransf}^t(\mathcal{H}^0, \mathcal{G}^{(t-1)}),\\ \mathcal{C}^t_H &= \mathbb{W}^{H,t}\mathcal{Z}^t + b^{H,t},\\ \mathcal{C}^t_D &= \mathbb{W}^{D,t}\mathcal{Z}^t + b^{D,t}.\end{aligned}$$

where $\textsc{GraphTransf}(\cdot)$ involves encoding the previous parsing graph to obtain new representations. In [Mohammadshahi and Henderson, 2021], it is implemented as a Recursive Non-autoregressive Graph-to-Graph Transformer. $\mathcal{C}^t_H$ and $\mathcal{C}^t_D$ are then scored into $\mathcal{S}^t$ and decoded to graph G_t as in the base biaffine parser.

$$\begin{aligned}\mathcal{S}^t &= \text{BIAF}^t(\mathcal{C}^t_H, \mathcal{C}^t_D),\\ \mathcal{G}_t &= \textsc{Decode}(\mathcal{S}^t).\end{aligned}$$

Higher-Order Representation-Driven (HORD) involves incorporating higher-order information, such as interdependencies between graph edges, into the graph representations for refinement to improve the base model's performance [Li *et al.*, 2020]. [Wang *et al.*, 2019] proposed a refiner with second-order features, which inducts based on second-order relationships among edges. The refiner uses Variational Inference (VI) for utilizing second-order representations into

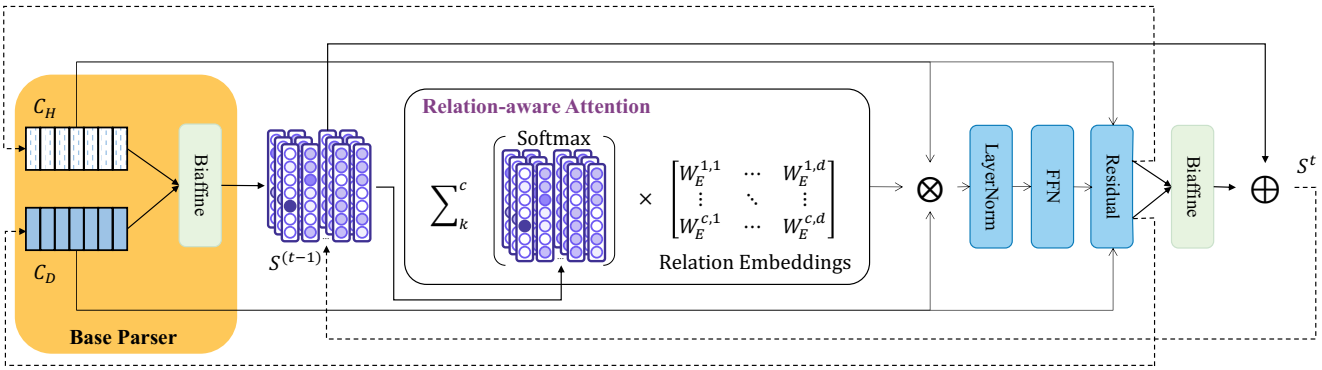

Figure 2: Process of our proposed retrospecting via message passing process. Dashed lines represent update operations during iterations.

the graph representations to further improve the final parsing performance.

Specifically, the second-order refiner predicts three second-order representations between words: siblings, co-parents, and grandparents. To get these representations, second-order refiner introduces a triaffine attention mechanism for scoring:

$$\mathcal{S}^{(bi)} = \text{TRIAF}(r_1, r_2, r_3) = {r_3}^T {r_1}^T \mathbb{U} r_2.$$

Similar to graph representations, second-order representations $\mathcal{V}^{sib}$, $\mathcal{V}^{cop}$ and $\mathcal{V}^{grp}$ are extracted on the role-specific sequence representations:

$$\begin{aligned}
\mathcal{V}^{(sib)} &= \text{TRIAF}^{(sib)}(\mathcal{C}^H, \mathcal{C}^D, \mathcal{C}^D), \\
\mathcal{V}^{(cop)} &= \text{TRIAF}^{(cop)}(\mathcal{C}^H, \mathcal{C}^D, \mathcal{C}^H), \\
\mathcal{V}^{(grp)} &= \text{TRIAF}^{(grp)}(\mathcal{C}^H, \mathcal{C}^B, \mathcal{C}^D),
\end{aligned}$$

where $\mathcal{C}^\lambda = \mathbb{W}_{so}^\lambda \mathcal{C} + b_{so}^\lambda, \lambda \in H, D, B$, and B indicates the role of both head and dependent. Then these second-order representations are used to refine the first-order graph representations by Mean Field Variational Inference [Wang *et al.*, 2019] to form final graph representations:

$$\mathcal{S}^t = \text{MFVI}(\mathcal{S}^0, \mathcal{V}^{(sib)}, \mathcal{V}^{(cop)}, \mathcal{V}^{(grp)}).$$

3.3 Retrospecting via Message Passing

In the methods SRD, JSSRD, and HORD, the sequence representation, joint sequence-structure representation, and high-order representation are used to guide the refinement of the graph representation, respectively, by re-encoding or additional scoring. However, this leads to a heavily increased computational cost relative to the base parser. To address this issue, we propose a new approach that uses message passing in graph representation learning to retrospect the draft graph representation, allowing for the update of the graph representation without the need for re-encoding or additional complex computations on the graph or sequence representation.

Specifically, the proposed retrospecting mechanism updates the graph representations by commuting between head-dependent related representations, allowing for a shortcut between related words based on the parsed graph in previous iterations. The process for refining graph representations using retrospective message passing involves taking head and dependent representations and parsing graph representations from the previous iteration as input. A relation-aware attention mechanism integrates the graph representations into relation-aware sequence representations, which involves a soft embedding process to transform relationship logits into relation features. Relation features are then fused with role-specific sequence features using element-wise product operations to obtain relation-aware sequence features, allowing head and dependent representations to interact with label features. Then, the message passing is finished by passing relation-aware sequence representations to inform about heads and dependents. The resulting relation-aware sequence representation is then used to reason a new graph representation, merged with the input graph representation using a shortcut mechanism to produce the output graph representation for the current iteration. The proposed approach enables considering the previous parsing graph when refining the current one.

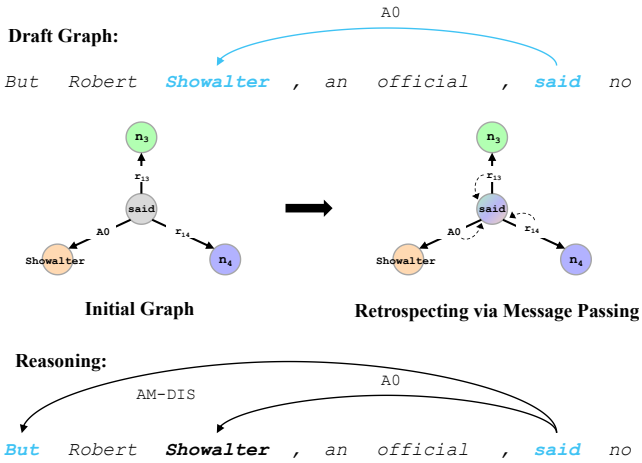

Figure 3: An instance showing how message passing works for parsing graph refinement. The "reasoning" process refers to the process of generating a new graph based on the draft parse and text input. It involves updating a node's representation by incorporating information from neighboring nodes in draft parses and refining the graph structure through iterative refinement.

For example, in the semantic role labeling task shown in

Figure 3, the shortcut informs the predicate '*said*' that it has an '*A0*' argument '*Showalter*' in the refining iteration. As a result, the predicate '*said*' is able to successfully reason the missing '*AM-DIS*' argument '*But*' thanks to the relationship between arguments '*But*' and the shortcut-identified '*Showalter*'. In other words, '*Showalter*' and '*But*' are parts of a sentence that have a local relationship, '*But*' serves to connect the clause containing '*Showalter*' with the implied contrasting clause. The shortcut mechanism updates the representation using the draft parse and establishes a connection between '*Showalter*' and '*said*'. The updated representation is subsequently used to further refine the tree in the next iteration, making it easier to reason about the connection between '*But*' and '*said*'.

In Figure 2, we illustrate the process for refining graph representations using retrospective message passing. In iteration t, our refinement takes the head and dependent representations, $C_H^{(t-1)}$ and $C_D^{(t-1)}$, as well as the parsing graph representations from the previous iteration, $S^{(t-1)}$, as input. We employ a relation-aware attention mechanism to integrate the graph representations into relation-aware sequence representations, which involves a soft embedding process to transform relationship logits into relationship features.

$$\mathcal{Q}_{i,j}^{(t-1)} = \textsc{Softmax}(\mathcal{S}_{i,j}^{(t-1)}),$$
$$\mathcal{E}_{i,j}^{(t-1)} = \sum_{k=1}^{c} \mathcal{Q}_{i,j,k}^{(t-1)} \mathcal{W}_E^k,$$

where i, j refers to the edge from i-th word to j-th word. $\mathcal{Q}_{i,j,k}$ thus represents the probability of the k-th label on the edge. $\mathcal{W}_E^k$ is the embedding for k-th label. The final output $\mathcal{E}_{i,j}^{(t-1)}$ is a sum of all label embeddings weighted by their existing probabilities on the edge.

We then fuse relation features with role-specific sequence features using element-wise product operations to obtain relation-aware sequence features. This allows head and dependent representations to interact with label features.

$$\mathcal{I}_{i,j}^{H,(t-1)} = \mathcal{E}_{i,j}^{(t-1)} \times \mathcal{C}_H^{(t-1),i},$$
$$\mathcal{I}_{i,j}^{D,(t-1)} = \mathcal{E}_{i,j}^{(t-1)} \times \mathcal{C}_D^{(t-1),j},$$

where $\times$ represents the element-wise product operation. $\mathcal{I}^H$, $\mathcal{I}^D$ represents the interaction results of relation features with their heads and dependents.

We further apply layer normalization, a feedforward network (FFN), and residual connection to the relation-aware sequence representations, as done in [Vaswani *et al.*, 2017], to improve the learning efficiency and reduce the risk of overfitting. These techniques enable the model to learn more complex relationships between the input and output, and improve optimization and generalization by learning the residual mapping between the input and output.

$$\mathcal{R}^{H,(t-1)} = \text{FFN}(\textsc{LayerNorm}(\mathcal{I}^{H,(t-1)})),$$
$$\mathcal{R}^{D,(t-1)} = \text{FFN}(\textsc{LayerNorm}(\mathcal{I}^{D,(t-1)})),$$
$$\mathcal{C}_H^t = C_H^{(t-1)} + \mathcal{R}^{H,(t-1)},$$
$$\mathcal{C}_D^t = C_D^{(t-1)} + \mathcal{R}^{D,(t-1)}.$$

Models	PTB		CTB	
	UAS	LAS	UAS	LAS
[Dozat and Manning, 2017]	95.74	94.08	89.30	88.23
[Ma *et al.*, 2018]	95.87	94.19	90.59	89.29
[Clark *et al.*, 2018]	96.60	95.00	-	-
[F & G, 2019]	96.04	94.43	-	-
[Ji *et al.*, 2019]	95.97	94.31	-	-
[Zhou and Zhao, 2019]	96.09	94.68	91.21	89.15
[Zhang *et al.*, 2020]	96.14	94.49	-	-
[Wang and Tu, 2020]	95.98	94.34	90.81	89.57
BIAF	96.62	94.97	91.66	89.49
BIAF+SRD	96.68	95.10	91.82	89.65
BIAF+JSSRD	96.66	95.01	91.88	89.66
BIAF+HORD	96.71	95.13	91.90	89.63
BIAF+IRE^{2}F	**96.85**	**95.31**	**92.04**	**89.88**
BIAF+HORD+SRD	96.53	95.02	91.75	89.60
BIAF+HORD+JSSRD	96.87	95.29	**92.14**	89.92
BIAF+HORD+IRE^{2}F	96.98	**95.60**	**92.13**	**89.97**

Table 1: Results on PTB-3.3 test set. Results in underline that the improvement of results over the BIAF baseline is significant ($\alpha = 0.01$).

After the above process, message passing is finished by passing $\mathcal{I}^H$ to inform $\mathcal{C}_D$ about its heads and passing $\mathcal{I}^D$ to $\mathcal{C}_H$ to inform about its dependents.

The resulting relation-aware sequence representation is then used to reason a new graph representation, merged with the input graph representation using a shortcut mechanism to produce the output graph representation for iteration t. This allows us to consider the previous parsing graph when refining the current one.

$$\tilde{\mathcal{S}}^t = \textsc{BIAF}(H^t, D^t),$$
$$\mathcal{S}^t = \textsc{Shortcut}(S^{(t-1)}, \tilde{\mathcal{S}}^t),$$

where H^t represents the sequence representation in the t-th iteration, H^0 denotes the input embeddings, and $\textsc{Shortcut}(\cdot)$ is implemented as element-wise addition.

4 Experiments

4.1 Datasets

We conducted experiments on three language structure prediction tasks: syntactic dependency parsing (SynDP), semantic dependency parsing (SemDP) and semantic role labeling (SRL).

SynDP We evaluate our proposed framework on traditional treebanks for English (PTB-3.3 [Marcus *et al.*, 1993]) and Chinese (CTB-5.1 [Xue *et al.*, 2002]) and also on the multilingual benchmark of universal dependency treebanks v2.3, in which 12 languages are selected following [Mohammadshahi and Henderson, 2021] for comparison with previous systems.

SemDP We conduct experiments on the datasets from SemEval-2015 shared task [Oepen *et al.*, 2015], which includes three subtasks: DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM), Enju Predicate–Argument Structures (PAS), and Prague Semantic Dependencies (PSD). Each subtask includes test sets in two domains: in-domain (ID) and out-of-domain (OOD).

Languages	BIAF	+SRD	+JSSRD	+IRE^{2}F
ar	85.55	85.88	85.95	**85.99**
eu	87.54	87.63	87.60	**87.86**
zh	88.87	89.02	89.32	**89.40**
en	90.75	90.80	91.29	**91.82**
fi	91.87	92.15	92.48	**92.58**
he	90.72	90.69	**91.27**	91.20
hi	94.21	94.25	94.18	**94.44**
it	94.68	94.70	**95.00**	94.97
ja	95.32	95.46	95.40	**95.59**
ko	86.73	87.25	**87.35**	87.26
ru	95.23	95.54	95.64	**95.93**
sv	91.68	91.86	92.05	**92.12**
tr	68.38	69.17	69.96	**70.16**
Avg.	89.34	89.57	89.80	**89.95**

Table 2: Performance comparison among baseline and different refiners on universal dependency treebanks v2.3. LAS metrics are reported in this table.

SRL In our experiments on SRL, we use the CoNLL09 shared task [Hajic *et al.*, 2009] which includes datasets in 7 languages [Li *et al.*, 2021].

4.2 Configurations

We use the hyperparameter setup from [Dozat and Manning, 2017] and [Dozat and Manning, 2018] for SynDP and SemDP, respectively. For English datasets, we use GloVe [Pennington *et al.*, 2014] as pre-trained word embeddings, and fasttext [Bojanowski *et al.*, 2017] for other languages. We also include lemma and part-of-speech features, which are embedded using randomly initialized embedding layer. For character features, we encode the sequence using a character BiLSTM. We use `bert-large-uncased` as the pre-trained language model for English and `bert-base-multilingual-cased` for other languages, except for Chinese which uses `bert-base-chinese`.

We implemented the SRD and JSSRD refiners based on the methods described in [Lyu *et al.*, 2019a] and [Mohammadshahi and Henderson, 2021]. For SynDP, we implemented the HORD refiner following the method in [Wang and Tu, 2020], for SemDP, we implemented it based on the method in [Wang *et al.*, 2019], and for SRL, we implemented it based on the method in [Li *et al.*, 2020]. The number of iterations in the refiners was determined through experiments on PTB for SynDP, PAS for SemDP, and CoNLL09-En for SRL.

In the refinement training, we use the sum of the loss between the golden edges and labels with scores from all iterations as the total loss. For SemDP and SRL tasks, we use binary cross-entropy loss for edges and cross-entropy loss for labels. For SynDP, we use cross-entropy loss for both edges and labels. We train the model using Adam optimizer [Kingma and Ba, 2015] with a decaying learning rate. The code is available at https://github.com/zcli-charlie/iRe2f.

4.3 Results

SynDP According to the results in Table 1 on SynDP, our baseline models performed well and were comparable to previously reported results. Comparing our IRE^{2}F with SRD, JSSRD and HORD, our IRE^{2}F achieves better refinement performance on the same strong BIAF baseline, illustrating the effectiveness of our method. The full model with IRE^{2}F achieved competitive results with previous best models[1] without stronger pre-trained language model and complex model designs. This demonstrates that our retrospecting refinement is a general and effective mechanism that can be applied to SynDP.

In addition, since the contributions of HORD and SRD, JSSRD, and IRE^{2}F are independent, we further combine them. The experimental results show that the model combining the two refinement mechanisms has been further improved compared to using a single method. This suggests that the improvement brought about by the refinement of SRD, JSSRD and IRE^{2}F does not solely depend on the high-order information present in the graph, but also on better graph representation learning.

Furthermore, we conducted experiments on universal dependency treebanks to verify the effectiveness of our refinement framework on multilingual scenarios. We compared the performance of IRE^{2}F with other iterative refiners, and the results are presented in Table 2. These experiments showed that our proposed IRE^{2}F framework outperforms other iterative refiners and is able to achieve superior results on multiple languages.

SDP Table 3 shows the comparison of our models with previous state-of-the-art methods. We first test the performance of the baseline biaffine model and various refinements including SRD, JSSRD, HORD, and IRE^{2}F. We find that all these refinements lead to performance gains and are comparable to previously reported results. However, based on the BIAF baseline, our IRE^{2}F framework brings about the biggest parsing performance improvement. Furthermore, our model can achieve further improvement on the stronger second-order baseline, which again indicates that IRE^{2}F refinement is orthogonal to high-order refinement. The results reached by our full model, which combines IRE^{2}F and HORD refinements, have set new state-of-the-arts on all SemDP datasets.

SRL We report in Table 4 the results on pre-identified predicates setting for 7 languages on the SRL task. BIAF+IRE^{2}F and BIAF+HORD have improved the parsing performance compared to the baseline BIAF, which reflects the importance of refinement in this task. Furthermore, superimposing IRE^{2}F on top of HORD shows even greater improvements in all seven languages compared to the BIAF baseline. Among them, *ca*, *es*, and *en* languages have the greatest improvement. This is because the proportion of second-order structures in these languages is small, so HORD brings limited improvement, and the effectiveness of our IRE^{2}F comes from graph representation learning, which can still play a role.

Computing Efficiency To demonstrate the computational efficiency advantages of IRE^{2}F compared to existing refinements, we selected the DM task in SemDP to compare model

[1]It is worth noting that the results reported in the table are based on the BERT pre-training model only, and stronger pre-training models have not been considered as they are not our focus.

Models	DM		PAS		PSD		Avg.	
	ID	OOD	ID	OOD	ID	OOD	ID	OOD
[Du *et al.*, 2015]	89.09	81.84	91.26	87.23	75.66	73.28	85.34	80.78
[Almeida and Martins, 2015]	88.21	81.75	90.88	86.88	76.36	74.82	85.15	81.15
[Peng *et al.*, 2017]	90.40	85.30	92.70	89.00	78.50	76.40	87.20	83.60
[Wang *et al.*, 2018]	90.30	84.90	91.70	87.60	78.60	75.90	86.90	82.80
BIAF [Dozat and Manning, 2018]	93.70	88.90	93.90	90.60	81.00	79.40	89.50	86.30
BIAF+HORD [Wang *et al.*, 2019]	93.90	89.50	94.20	91.30	81.40	79.50	89.80	86.80
BIAF	94.61	91.59	95.04	93.04	82.55	80.38	90.73	88.34
BIAF+SRD	94.56	91.80	95.03	93.12	82.60	80.35	90.73	88.42
BIAF+JSSRD	95.01	91.72	95.19	93.33	82.96	80.59	91.05	88.55
BIAF+HORD	94.87	91.80	95.35	93.70	83.07	80.71	91.10	88.74
BIAF+IRE^{2}F	95.02	91.94	95.63	93.94	83.05	80.82	91.23	88.90
BIAF+HORD+IRE^{2}F	**95.43**	**92.06**	**95.52**	**94.01**	**83.52**	**80.90**	**91.49**	**89.02**

Table 3: Comparison of results on SemEval-2015 test sets for DM, PAS, PSD subtasks.

Models	*ca*	*cs*	*de*	*en*	*es*	*ja*	*zh*	Avg.
CoNLL-2009 ST	80.3	85.4	79.7	85.6	80.5	78.2	78.6	81.19
[Zhao *et al.*, 2009]	80.3	85.2	76.0	86.2	80.5	78.2	77.7	80.59
[Marcheggiani *et al.*, 2017]	–	86.0	–	87.7	80.3	–	81.2	–
[Mulcaire *et al.*, 2018]	79.45	85.14	69.97	87.24	77.32	76.00	81.89	79.57
[Lyu *et al.*, 2019b]	80.91	87.62	75.87	90.99	80.53	82.54	83.31	83.11
[He *et al.*, 2019]	85.14	89.66	80.87	90.86	84.60	83.76	86.42	85.90
BIAF [Li *et al.*, 2020]	86.40	91.48	85.21	91.23	86.60	85.55	88.24	87.82
BIAF+HORD [Li *et al.*, 2020]	86.90	91.93	85.54	91.77	86.96	85.90	88.69	88.24
BIAF+IRE^{2}F	87.15	91.90	85.68	92.03	87.35	85.95	88.57	88.38
BIAF+HORD+IRE^{2}F	**87.40**	**92.16**	**85.93**	**92.11**	**87.52**	**86.10**	**88.87**	**88.58**

Table 4: Results on the CoNLL-2009 in-domain test sets. Semantic-F_1 score is the evaluation metric.

Models	Params		Efficiency	
	Base	*Refine*	*Train* (↑)	*Infer* (↑)
BIAF	161.98M	–	1.02	186.2
BIAF+SRD	161.98M	133.28M	0.71	138.6
BIAF+JSSRD	161.98M	152.10M	0.52	98.0
BIAF+HORD	161.98M	19.42M	0.83	140.9
BIAF+IRE^{2}F	161.98M	13.23M	0.95	152.5

Table 5: Comparison of the parameters and efficiencies of models without and with different refinements. In efficiency, the metric for *Train* is epoch/hour, and for *Infer* is sentences/second.

parameters and efficiency. The results are shown in Table 5. The models with +SRD, +JSSRD, +HORD, +IRE^{2}F have consistent base parameters with the BiAF baseline, and the introduction of refinement brings additional parameters. Among them, IRE^{2}F has the least additional parameters, about 90% less than the previous minimum SRD, which is related to its mechanism of directly updating the graph representation based on retrospecting without re-encoding. Correspondingly, IRE^{2}F has the least decrease in training and inference speed compared to the baseline, while JSSRD has the most decrease, which may be due to the need to perform additional MFVI in high-order reasoning. By comparing with +SRD, +JSSRD, +HORD, we conclude that IRE^{2}F is an efficient refinement framework.

5 Further Analysis

5.1 Ablation Study

We present an ablation study of the structural design of IRE^{2}F in Table 6. After removing the SHORTCUT, the parsing performance drops significantly, even lower than the BIAF baseline. This illustrates that in the iterative refinement process of IRE^{2}F, it is crucial to update the graph representation gradually, which can reduce the difficulty of graph representation learning. Removing RELATTN also resulted in a decrease in performance, which demonstrates that the graph representation is not only dependent on the edges, but the relationship between edges is also crucial for refinement. The removal of FFN and RESIDUAL makes IRE^{2}F unable to be successfully trained, indicating that it is helpful to effectively learns complex representations and overcomes the limitations like vanishing gradient problem.

5.2 Different Iterations

IRE^{2}F performs iterative refinement of graph representations. We show in Table 7 the effect of the number of iterations on the parsing performance of SynDP and SemDP tasks. Compared to no refinement, even one round of refinement can achieve better results. Further increasing the number of iterative rounds leads to an improvement in parsing performance. However, the optimal number of iterations for different tasks varies depending on the complexity of the parsing graph. The

Settings	SemDP (PAS)		Params	Speed
	ID	OOD	*Refine*	(Sent/Sec)
BIAF	95.04	93.04	–	186.2
BIAF+IRE^{2}F	95.63	93.94	13.23M	152.5
- SHORTCUT	93.74	92.01	13.23M	153.8
- RELATTN	95.39	93.26	2.95M	168.4
- FFN	95.46	92.98	11.75M	159.7
- RESIDUAL	52.44	31.63	13.23M	154.1

Table 6: Ablation study of SHORTCUT, Relation-aware Attention (RELATTN), FFN and Residual connection (RESIDUAL) structures in IRE^{2}F.

Iter.	SynDP (PTB)		SemDP (PAS)	
	UAS	LAS	ID	OOD
1	96.75	95.19	95.12	93.29
2	96.70	95.22	95.29	93.52
3	96.81	95.26	95.27	93.56
4	96.76	95.19	**95.33**	**93.64**
5	**96.85**	**95.31**	95.23	93.51
6	96.81	95.24	95.27	93.48

Table 7: Influence of number of refine iterations for the performance of IRE^{2}F.

number of iterative rounds should not be as high as possible. After the effect of refinement and improvement reaches a the optimal, continuing to increase will not bring about any further improvement or may even result in a decline.

5.3 More Discussion for Longer Sentences

In the iterative refinement method, both [Lyu *et al.*, 2019a] and [Mohammadshahi and Henderson, 2021] have compared the performance of using iterative refinement on different dependency distances, and it is indeed true that iterative refinement can improve the parsing accuracy of instances with longer dependency distances. Additionally, we also conducted separate statistics on the UD v2.3 test set based on sequences of different lengths. From the statistical results shown in Figure 4, there is indeed a trend of greater improvement for longer sequences, but it is not a perfect match. On the one hand, sequences with longer lengths have greater dependency distances, but they are not entirely consistent. On the other hand, sequences of different lengths have different proportions in the test set, which can lead to bias in certain length intervals.

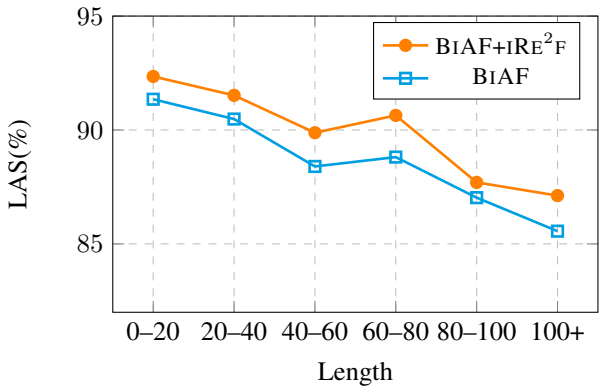

Figure 4: performance on different length spans.

6 Conclusion and Future Work

In this paper, we presented an efficient refinement framework, IRE^{2}F, for language structure prediction tasks. IRE^{2}F performs refinement by iteratively retrospecting and reasoning on the graph representations directly, without the need for re-encoding. By introducing the novel relation-aware attention, our structure achieved significant improvement. Furthermore, our proposed IRE^{2}F can also be combined with other refinement frameworks like HORD for further improvement. Through various experiments, we demonstrated that IRE^{2}F outperforms existing state-of-the-art refinement frameworks in terms of performance and computational efficiency. Since our refinement approach is orthogonal to the choice of base parser, we evaluated our proposed method using only the biaffine parser. This parser is widely used and is a strong foundation for graph parsing. However, we anticipate that our approach can be applied to other base parsers as well, which would demonstrate its generalization ability for future work. We also plan to investigate its performance with other base parsers.

Acknowledgements

Corresponding author: Hai Zhao. Zuchao Li and Letian Peng made equal contributions to this work. This work was supported by the Special Fund of Hubei Luojia Laboratory under Grant 220100014. Hai Zhao was funded by the Key Projects of National Natural Science Foundation of China (U1836222 and 61733011). We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

[Almeida and Martins, 2015] Mariana S. C. Almeida and André F. T. Martins. Lisbon: Evaluating turbosemantic-parser on multiple languages and out-of-domain data. In Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 970–973, 2015.

[Bocharova, 2008] V Bocharova. Rule-based language structure prediction. *Natural Language Engineering*, 14(3):299–321, 2008.

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[Chen and Manning, 2014] Dan Chen and Christopher D Manning. A fast and accurate dependency parser using

neural networks. In *Proceedings of EMNLP*, pages 740–750, 2014.

[Clark *et al.*, 2018] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, October-November 2018.

[Dozat and Manning, 2017] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[Dozat and Manning, 2018] Timothy Dozat and Christopher D. Manning. Simpler but more accurate semantic dependency parsing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 484–490, 2018.

[Du *et al.*, 2015] Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. Peking: Building semantic dependency graphs with a hybrid parser. In Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of SemEval@NAACL-HLT*, pages 927–931, 2015.

[Fernández-González and Gómez-Rodríguez, 2019] Daniel Fernández-González and Carlos Gómez-Rodríguez. Left-to-right dependency parsing with pointer networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 710–716, Minneapolis, Minnesota, June 2019.

[Gui *et al.*, 2020] Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. Uncertainty-aware label refinement for sequence labeling. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2316–2326, 2020.

[Hajic *et al.*, 2009] Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Jan Hajic, editor, *Proceedings of the CoNLL*, pages 1–18, 2009.

[Hashimoto *et al.*, 2016] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.01587, 2016.

[He *et al.*, 2019] Shexia He, Zuchao Li, and Hai Zhao. Syntax-aware multilingual semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5350–5359, November 2019.

[Hua and Wang, 2020] Xinyu Hua and Lu Wang. PAIR: planning and iterative refinement in pre-trained transformers for long text generation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of EMNLP*, pages 781–793, 2020.

[Ji *et al.*, 2019] Tao Ji, Yuanbin Wu, and Man Lan. Graph-based dependency parsing with graph neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2475–2485, July 2019.

[Jia *et al.*, 2020] Zixia Jia, Youmi Ma, Jiong Cai, and Kewei Tu. Semi-supervised semantic dependency parsing using CRF autoencoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6795–6805, July 2020.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Kiperwasser and Goldberg, 2016] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Trans. Assoc. Comput. Linguistics*, 4:313–327, 2016.

[Kurita and Søgaard, 2019] Shuhei Kurita and Anders Søgaard. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy, July 2019.

[Lee *et al.*, 2020] Jason Lee, Raphael Shu, and Kyunghyun Cho. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of EMNLP*, pages 1006–1015, 2020.

[Li *et al.*, 2020] Zuchao Li, Hai Zhao, Rui Wang, and Kevin Parnow. High-order semantic role labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1134–1151, November 2020.

[Li *et al.*, 2021] Zuchao Li, Hai Zhao, Shexia He, and Jiaxun Cai. Syntax role for neural semantic role labeling. *Comput. Linguistics*, 47(3):529–574, 2021.

[Li *et al.*, 2022a] Zuchao Li, Zhuosheng Zhang, Hai Zhao, Rui Wang, Kehai Chen, Masao Utiyama, and Eiichiro Sumita. Text compression-aided transformer encoding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3840–3857, 2022.

[Li *et al.*, 2022b] Zuchao Li, Junru Zhou, Hai Zhao, Zhisong Zhang, Haonan Li, and Yuqi Ju. Neural character-level syntactic parsing for chinese. *J. Artif. Intell. Res.*, 73:461–509, 2022.

[Lyu *et al.*, 2019a] Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. Semantic role labeling with iterative structure refinement. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of EMNLP-IJCNLP*, pages 1071–1082, 2019.

[Lyu *et al.*, 2019b] Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. Semantic role labeling with iterative structure refinement. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1071–1082, November 2019.

[Ma *et al.*, 2018] Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 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, July 2018.

[Marcheggiani *et al.*, 2017] Diego Marcheggiani, Anton Frolov, and Ivan Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, August 2017.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Mohammadshahi and Henderson, 2021] Alireza Mohammadshahi and James Henderson. Recursive non-autoregressive graph-to-graph transformer for dependency parsing with iterative refinement. *Trans. Assoc. Comput. Linguistics*, 9:120–138, 2021.

[Mulcaire *et al.*, 2018] Phoebe Mulcaire, Swabha Swayamdipta, and Noah A. Smith. Polyglot semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 667–672, July 2018.

[Nivre *et al.*, 2009] Joakim Nivre, Wim Peters, and Ivan Titov. Statistical dependency parsing: An introduction. *Synthesis Lectures on Human Language Technologies*, 2(1):1–175, 2009.

[Oepen *et al.*, 2015] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In Daniel M. Cer, David Jurgens, Preslav Nakov, and Torsten Zesch, editors, *Proceedings of SemEval@NAACL-HLT*, pages 915–926, 2015.

[Peng *et al.*, 2017] Hao Peng, Sam Thomson, and Noah A. Smith. Deep multitask learning for semantic dependency parsing. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of ACL*, pages 2037–2048, 2017.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, October 2014.

[Sun *et al.*, 2021] Kailai Sun, Zuchao Li, and Hai Zhao. Multilingual pre-training with universal dependency learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS*, pages 8444–8456, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, pages 5998–6008, 2017.

[Wang and Tu, 2020] Xinyu Wang and Kewei Tu. Second-order neural dependency parsing with message passing and end-to-end training. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of AACL/IJCNLP*, pages 93–99, 2020.

[Wang *et al.*, 2018] Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. A neural transition-based approach for semantic dependency graph parsing. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of AAAI*, pages 5561–5568, 2018.

[Wang *et al.*, 2019] Xinyu Wang, Jingxian Huang, and Kewei Tu. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, July 2019.

[Xue *et al.*, 2002] Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. Building a large-scale annotated Chinese corpus. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[Zhang *et al.*, 2020] Yu Zhang, Zhenghua Li, and Min Zhang. Efficient second-order treecrf for neural dependency parsing. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of ACL*, pages 3295–3305, 2020.

[Zhao *et al.*, 2009] Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of CoNLL*, pages 61–66, 2009.

[Zhou and Zhao, 2019] Junru Zhou and Hai Zhao. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, July 2019.