Dependency Parsing with the Structuralized Prompt Template

Anonymous ACL submission

Abstract

Dependency parsing is a fundamental task in natural language processing (NLP), aiming to identify syntactic dependencies and construct a syntactic tree for a given sentence. Traditional dependency parsing models typically construct embeddings and utilize additional layers for prediction. We propose a novel dependency parsing method that relies solely on an encoder model with a text-to-text training approach. To facilitate this, we introduce a structured prompt template that effectively captures the structural information of dependency trees. Our method achieves the state-of-the-art performance in UAS (97.41) and outperforms most previous approaches in LAS (96.16) on the English Penn Treebank, despite relying solely on a pre-trained model. Furthermore, this method is highly adaptable to various pre-trained models across different target languages and training environments, allowing easy integration of task-specific features.[1]

1 Introduction

Dependency parsing is a fundamental task in natural language processing (NLP) that analyzes syntactic relationships between words in a sentence. Figure 1 illustrates the traditional dependency parsing pipeline. Traditionally, dependency parsing is performed in two steps: (1) creating word-level embeddings, and (2) identifying the head word of each word along with its dependency relation using the generated embeddings.

Previously, dependency parsing primarily relied on simple pre-processed contextual vectors to initialize embeddings (Li et al., 2018; Strzyz et al., 2019; Vacareanu et al., 2020). With the advent of powerful pre-trained language models such as BERT (Devlin et al., 2019), recent dependency parsing approaches leverage these models to initialize word embeddings, achieving superior performance compared to earlier methods (Amini et al., 2023). In the second step of dependency parsing, previous studies have shown that graph-based methods, such as biaffine (Dozat and Manning, 2017), yield good performance in identifying relations. Consequently, this approach was extended to learn the subtree information of the dependency tree (Yang and Tu, 2022). Since the structural characteristics of dependency trees increased training complexity, some studies use the sequence tagging method for parsing (Li et al., 2018; Amini and Cotterell, 2022). These approaches add layers after embedding construction and label the words into structural information sequentially. In particular, the hexatagging achieved SOTA performance by generating structural information with a finite set of tags through decoding (Amini et al., 2023). In addition, Lin et al. (2022) demonstrated that encoder-decoder models effectively generate relation unit texts from input, highlighting that parsing can be performed solely using pre-trained models. This study is particularly significant as it explores dependency parsing by transforming modified text into

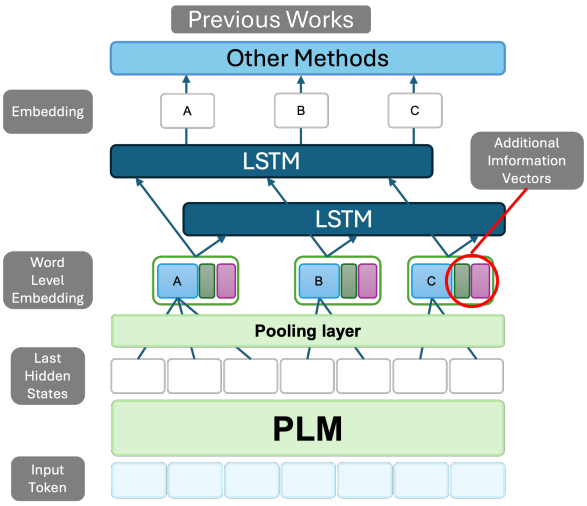


Figure 1: A figure shows that pipeline of traditional dependency parsing

[1] https://anonymous.4open.science/r/SPT-DP-C2C1

structured dependency output. However, the limitation of this approach lies in the increased computational time required for parsing, as the number of tokens grows due to the generation-based method.

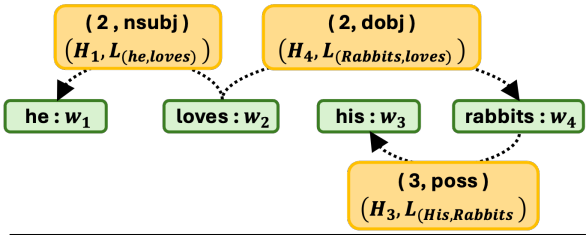
In this paper, we propose a novel method to perform dependency parsing solely on pre-trained encoder models that are constructed by prompt engineering using additional tokens as soft prompts. We hypothesize that prompt engineering can effectively convert the text-to-structure task in dependency parsing to the text-to-text task by pre-trained language models. Hence, the output text sequence of the proposed method has to reflect the tree structure of dependency parsing well. To achieve this, we design several soft prompts so that our model can identify the structural information of the tree structure, and then apply the **S**tructuralized **P**rompt **T**emplate (SPT) for each processing unit of dependency parsing using the developed soft prompt. We believe that prompt learning with the structuralized prompt template enables effective and efficient dependency parsing only on the pre-trained language models. Eventually, by learning through the structuralized prompt template, the **S**tructuralized **P**rompt **T**emplate based **D**ependency **P**arsing (SPT-DP) method achieves the high performance and simplified training by reducing the gap between pre-training and fine-tuning because it is based on only the pre-trained language models for the text-to-text task. As a result, our method achieves the state-of-the-art (SOTA) performance in UAS **(97.41)** and outperforms most existing approaches in LAS **(96.16)** on the English Penn Treebank (PTB; Marcus et al. (1993)). On the 2.2 version of Universal Dependencies (UD 2.2; Nivre et al. (2018)), it obtains the SOTA performance in 1 languages out of 12 languages when using the cross-lingual Roberta model (Liu et al., 2019). Since our method utilizes only a single encoder model, our approach is faster than existing methods. In particular, it achieves a 40x speed improvement compared to DPSG, which employs a similar text-to-text approach. Furthermore, our method achieves a performance comparable to that of the SOTA model with a complicated and heavy architecture in the Korean Sejong dataset.

2 Preliminaries

2.1 Dependency Parsing

Dependency parsing is the process of identifying dependency relationships among words in a sentence. Given a sentence $S = (w_1, w_2, \ldots, w_n)$, the task is to derive the dependency relations, $R_S = (r_1, r_2, \ldots, r_n)$, for each word. Each relation, $r_i = (H_i, L_{(w_i, w_{H_i})})$, consists of two components: (1) the index of the head word H_i and (2) the dependency relation label $L_{(w_i, w_{H_i})}$, between w_i and w_{H_i}, where $L_{(w_i, w_{H_i})} \in \mathcal{L}$ and $\mathcal{L}$ is the set of predefined dependency relation labels. This study focuses on syntactic dependency parsing that analyzes grammatical dependency relations within sentences. Depending on the definition of the syntactic dependency tree, each word has exactly one parent node, ensuring a hierarchical structure. Therefore, the ultimate goal of dependency parsing is to analyze the sentence S into the form $S_{dep} = \{(w_i, r_i) \mid r_i = (H_i, L_{(w_i, w_{H_i})}), i \in \{1, 2, \ldots, n\}\}$, where each word w_i is paired with its corresponding dependency relation r_i. [2]

$$S = (He, loves, his, rabbits)$$



Word : w_i	**Index** : i	**Head** : H_i	**Label** : $L_{(w_i, w_{H_i})}$
he	1	2	[nsubj]
loves	2	0(root)	[root]
his	3	4	[poss]
rabbits	4	2	[dobj]

Figure 2: Dependency Parsing Example

2.2 Prompt based tuning

Initially, the concept of a prompt emerged as a method for training models, particularly to bridge the gap between the pre-training and fine-tuning stages. PET (Pattern-Exploiting Training) proposed a method to reduce the gap between pre-training and fine-tuning by using cloze-style patterns instead of the traditional [CLS] token-based classification approach in BERT (Schick and Schütze, 2021). This allows the pre-trained language model to naturally adapt to classification tasks. For example, in sentiment analysis, a sentence like "This movie is really [MASK]." is used, and the predicted probability of the [MASK] token is used to determine its sentiment.

Subsequently, prompts have been explored in

[2] Table 11 for a list of notations along with explanations.

combination with methods like PEFT (Parameter-Efficient Fine-Tuning) to improve model efficiency and task performance (Li and Liang, 2021; Lester et al., 2021). In generative models like GPT-3 (Brown et al., 2020), text-based prompts are used to guide and shape the outputs into task-specific outputs.

In this study, prompts are used to provide additional linguistic hints, such as grammatical structures or contextual information, necessary for dependency parsing. Specifically, prompts are added to the encoder model through preprocessing. This is done by concatenating prompt phrases, such as task-specific instructions, to the original input sentence. Prompts are applied to each word in the input sentence as shown in the following formulation:

$$P(S) = (T(w_1), T(w_2), \ldots, T(w_n)) \quad (1)$$

where $P(S)$ represents the transformed sentence and $T(w_i)$ is the prompt template applied to the i-th word w_i.

2.3 Pre-trained Language Model

Transformer-based pre-trained models are built upon the following architectures:

- **Encoder-based Models**: These models encode input text to perform tasks like masked token prediction or classification by adding a linear layer. They are also widely used in *Dense Passage Retrieval (DPR)* (Karpukhin et al., 2020) to extract representative embeddings for documents. $X \rightarrow X'$

- **Decoder-based Models**: These models take an initial token or prompt and generate tokens sequentially through autoregressive decoding. $X \rightarrow Y$

The traditional approach to dependency parsing involves extracting token-level embeddings from encoder-based pre-trained models, integrating them at the word level, and constructing word embeddings. These embeddings are then analyzed through various methods to perform dependency parsing. In this process, the performance of dependency parsing heavily depends on how effectively the encoder provides information relevant to syntactic structures.

3 Structuralized Prompt Template

As aforementioned, traditional methods approach dependency parsing by utilizing predefined embeddings processed through specific modules. However, Dependency Parsing via Sequence Generation (DPSG) departed from this paradigm and introduced a text-to-text dependency parsing method (Lin et al., 2022). Despite its innovative nature, DPSG faced limitations due to its generation-based approach, which could produce unintended outputs. In addition, the need to generate tokens one by one resulted in slower processing speeds, especially when processing additional tokens. To overcome these challenges, we propose a novel approach that leverages encoder models. Inspired by the ability of encoders to facilitate text-to-text learning through pre-training tasks such as Masked Language Modeling (MLM), we devised a method to represent dependency parsing information as text and apply it during training. We call this method **S**tructuralized **P**rompt **T**emplate (SPT). Unlike traditional dependency parsing, which treats text and dependency structures as separate levels, SPT integrates dependency information directly into text sequences. This approach enables dependency parsing entirely within the textual level by eliminating additional integration steps and improving overall efficiency.

3.1 Dependency Parsing with Text Representation

Basically, dependency parsing is the task of finding a word with a dependency relation and determining its corresponding dependency relation label for each word. To express the structured dependency relationships through the prompt template for each word, several conditions should be satisfied: 1) each template must be distinguished by its pattern through whole training, 2) it must be able to indicate its index, 3) it must be able to refer to the other word template with dependency relationship through the output, and 4) it must be able to express the dependency relation label through the output.

In the first condition, we ensure that the language model can distinctly recognize each template by following a consistent pattern rather than a specific token through all the training process. To satisfy the second and third conditions, we add index prompts P_{idx} that serve two roles: representing the template and indicating the referred template's index. Because the P_{idx} prompts represent the index of templates in a formatted text sequence, each

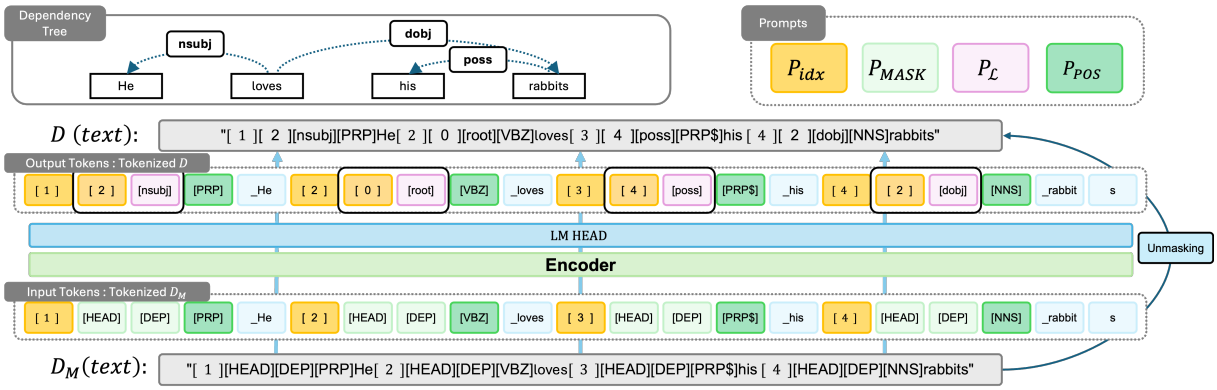


Figure 3: Overview of the Structuralized Prompt Template based Dependency Parsing (SPT-DP) method.

template can refer to another template within a dependency relation regardless of the input sequence using the P_{idx} prompts. Finally, the fourth condition is resolved by adding the dependency relation labels as prompt $P_{\mathcal{L}}$. In addition, we add P_{POS} to reflect information for part-of-speech (POS) analysis. However, several challenges arise when using this prompt approach for dependency parsing:

1. **Semantic Representation Issues**: If the prompts are directly integrated into the model's existing vocabulary without introducing new tokens, it may disrupt the original vocabulary structure, making it difficult to capture meaningful semantics.

2. **Variability in Length**: The explicit addition of prompts, such as numerical indices and dependency relation labels, causes variations in the length of the enhanced sentence. Masking these components can lead to further inconsistencies in sequence length, which is critical for learning in encoder models.

To address these issues, we propose to add each prompt as a new token to the vocabulary. The above prompts are individually added to the vocabulary enclosed in []. This action allows them to be added independently to the existing vocabulary without any duplications. This approach allows the model's training process to be examined at the text level. To avoid confusion, we clarify that the notations P_{idx} , $P_{\mathcal{L}}$, P_{POS} , and P_{MASK}[3] represent string text encapsulated in square brackets [].

$$T(w_i) = \text{“}\underbrace{\overbrace{[i]}^{P_{idx}}\overbrace{[H_i]}^{P_{idx}}}_{P_{abs}\ P_{ref}}\overbrace{[L_{(w_i,w_{H_i})}]}^{P_{\mathcal{L}}}\overbrace{[POS_{w_i}]}^{P_{POS}}\ w_i\text{”} \quad (2)$$

$$D = (T(w_1), T(w_2), ..., T(w_n)) \quad (3)$$

The final word-level prompt template is represented by Equation 2: for each word w_i, four prompts are added. In the first prompt $[i]$ is P_{abs}, which represents the absolute index token indicating the index of each word in the sentence. The second prompt $[H_i]$ is P_{ref} that specifies the index of the word referenced by the given word w_i. The only distinction between P_{abs} and P_{ref} is their positions in the template; they serve different roles but share the same tokens. That is why they are grouped under P_{idx}. The third is the dependency relation label $[L_{(w_i,w_{H_i})}]$, and the fourth contains $[POS_{w_i}]$ that is POS-tag of w_i. The proposed **S**tructuralized **P**rompt **T**emplate (SPT) can effectively incorporate the dependency parsing information into input sentence using these prompts. That is, by applying this template to every word in the sentence, we construct a modified sentence D that encapsulates dependency parsing information. From a certain perspective, performing dependency parsing can be regarded as transforming the original sentence S into a Prompted sentence D.

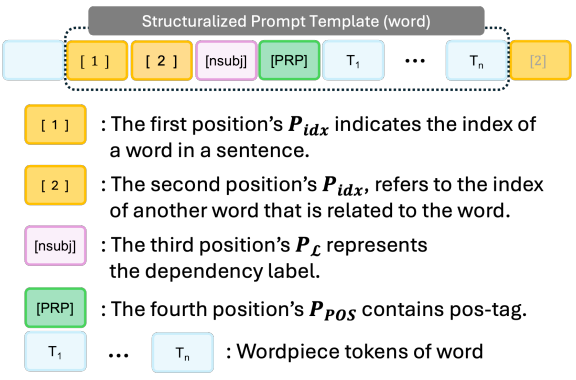


Figure 4: A figure of word level SPT example

[3]to be explained in Section 3.2.

3.2 Prediction Task using Soft Prompts P_{MASK} : [HEAD] and [DEP]

For the training, P_{MASK}, which contains the [HEAD] and [DEP] tokens, serves as a soft prompt and plays a role similar to the masked token in the Masked Language Model (MLM) task. Since performing dependency parsing involves predicting $[H_i]$ and $[L_{(w_i,w_{H_i})}]$, the corresponding parts in the equation are masked to enable the model to learn by making predictions. In this process, $[H_i]$ is masked as [HEAD] and $[L_{(w_i,w_{H_i})}]$ is masked as [DEP], respectively. The P_{MASK} are also added in the tokenizer's vocab for prompt engineering.

$$M(T(w_i)) = \text{“}\overbrace{[i]}^{P_{abs}}\overbrace{[HEAD][DEP]}^{P_{MASK}}[POS_{w_i}]w_i\text{”} \quad (4)$$

$$D_M = (M(T(w_1)), M(T(w_2)), ..., M(T(w_n))) \quad (5)$$

As aforementioned, the P_{MASK} are used to infer two main prediction tasks for the head word and the dependency relation label. They are arranged in the second and third positions of the structuralized prompt templates, forming a consistent pattern in the input text sequence. In our approach, the model is fine-tuned to predict head word and dependency relation labels by the [HEAD] and [DEP] prompts.

Ultimately, the goal is to learn to reconstruct the D from the masked input D_M, which has been structured using prompts. This involves different prediction mechanisms for encoder and decoder models:

Encoder-only Models The encoder model learns by reconstructing the masked input D_M into its original form D. Since the output sequence retains the same structure as the input sequence, the model effectively maps D_M back to D: $Enc(D_M) \rightarrow D$

Decoder-based Models In encoder-decoder models, the encoder processes the masked input D_M to generate a latent representation and the decoder uses it to reconstruct D. On the other hand, D_M of decoder-only models serves as a prefix and it guides the decoder to generate the unmasked D, autoregressively: $Gen(D_M) \rightarrow D$

4 Prompt-based Training

As mentioned earlier, in the encoder model, training is conducted in an MLM manner, where the input masked with P_{MASK}:[HEAD] and [DEP] are used to predict reference indices P_{ref}:$[H_i]$ and dependency relation label $P_{\mathcal{L}}$:$[L_{(w_i,w_{H_i})}]$ prompts. Since we have added each prompt to the tokenizer, the model is trained to output the corresponding prompt IDs through the LM head. In the decoder model, the training process involves providing masked input and learning to generate the unmasked text.

The loss function for training is calculated by the following equations. In Equation 6, X_{input} is the tokenized D_M that is a concatenated sequence of SPTs where $[H_i]$ and $[L_{(w_i,w_{H_i})}]$ are replaced by [HEAD] and [DEP] prompts. In Equation 7, Y_{label} is the tokenized D that is the prompted sequence based on SPT.

Since all the prompts are added to the tokenizer's vocabulary, the lengths of X_{input} and Y_{label} are always the same, which is a crucial condition in encoder-only models. In summary, the models are trained with D_M as input and D as the label. Encoder-only models are trained with $\mathcal{L}_{enc}$, which optimizes token prediction based on X_{input} following BERT's masked language modeling (MLM) for bidirectional contextual learning. Decoder-based models use $\mathcal{L}_{dec}$, where each token y_i is generated sequentially based on X_{input} and previous outputs $Y_{<i}$, following an autoregressive approach similar to GPT. θ is the parameter of model.

The objective function is based on **Cross-Entropy Loss**, defined as follows:

$$Tokenize(D_M) = X_{input} = [x_1, x_2, ..., x_N] \quad (6)$$

$$Tokenize(D) = Y_{label} = [y_1, y_2, ..., y_N] \quad (7)$$

$$\mathcal{L}_{enc} = -\sum_{i=1}^{N} \log P(y_i|X_{input}; \theta) \quad (8)$$

$$\mathcal{L}_{dec} = -\sum_{i=1}^{N} \log P(y_i|X_{input}, Y_{<i}; \theta) \quad (9)$$

In summary, $\mathcal{L}_{enc}$ and $\mathcal{L}_{dec}$ correspond to Cross-Entropy Loss, where the model optimizes token prediction probabilities. In encoder-only models, $\mathcal{L}_{enc}$ applies token-wise classification in an MLM setting, while in decoder-based models, $\mathcal{L}_{dec}$ follows an autoregressive generation paradigm.

5 Experiments

We first apply our approach to two datasets used in previous studies, including **PTB** (Penn Treebank) and **UD 2.2** (Nivre et al., 2018) covering 12 languages. Since these datasets primarily consist of **fusional** languages in which a single morpheme can

	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	Avg.
Dozat and Manning (2017)♢	90.30	**94.49**	92.65	**85.98**	91.13	93.78	**91.77**	94.72	91.04	94.21	87.24	94.53	91.82
Wang and Tu (2020)♢	91.30	93.60	92.09	82.00	90.75	92.62	89.32	93.66	91.21	91.74	86.40	92.61	90.61
Yang and Tu (2022)♢	91.10	94.46	92.57	85.87	91.32	**93.84**	91.69	**94.78**	91.65	94.28	87.48	94.45	91.96
Lin et al. (2022)*	**93.92**	93.75	92.97	84.84	**91.49**	92.37	90.73	94.59	**92.03**	**95.30**	**88.76**	**95.25**	**92.17**
Amini et al. (2023)♢	92.87	93.79	92.82	85.18	90.85	93.17	91.50	94.72	91.89	93.95	87.54	94.03	91.86
SPT-DP (XLM-RoBERTa-large)	92.63	92.90	**94.12**	84.53	90.55	92.13	91.32	93.63	91.69	93.17	88.60	94.98	91.63
SPT-DP (global)	93.69	93.14	93.23	84.02	90.12	92.52	91.20	93.64	91.98	92.97	88.73	94.27	91.63

Table 1: 12 languages' LAS scores on the test sets in UD 2.2. ♢ use multilingual BERT for embedding and * uses T5-base model for sequence generation parsing. Best and second-best scores are in **bold** and underlined

	PTB	
Model	UAS	LAS
Zhou and Zhao (2019)*	*97.0*	*95.43*
Mrini et al. (2020)*	*97.42*	*96.26*
Dozat and Manning (2017)	95.74	94.08
Wang and Tu (2020)	96.91	95.34
Yang and Tu (2022)	97.24	95.73
Lin et al. (2022)	96.64	95.82
Amini et al. (2023)	97.4	**96.4**
SPT-DP (XLNet-large)	**97.41**	96.16

Table 2: Results on PTB dataset. * use additional constituency parsing information so they are not comparable to other methods.

encode multiple grammatical features, we extended our experiments to **agglutinative** languages. Unlike fusional languages, agglutinative languages express grammatical relationships through sequences of distinct morphemes, which are carrying out their individual function. For this, we select the **Sejong** corpus, a Korean language dataset, as an agglutinative language dataset.

5.1 Datasets

PTB This English data is preprocessed by Stanford Parser v3.3.0 (de Marneffe and Manning, 2008) to convert it into CoNLL format, following the approach of Mrini et al. (2020).

UD 2.2 This is composed of 12 languages from UD dataset v2.2 and we follow previous work (Amini et al., 2023) for data splitting and organizing. The POS tag information is not used for the experiments by omitting P_{POS} in the template.

Sejong This is the Korean dataset and only the POS tags of the first and last morphemes are used for this experiment because Korean words consist of multiple morphemes.

5.2 Pre-trained Language Models

Encoder-based Models XLNet-large, Multilingual BERT, XLM-RoBERTa-large, and RoBERTa

Decoder-based Models T5-base, BART-large, LLaMA3.2-3B, and Qwen2.5-3B

5.3 Comparison Models

Zhou and Zhao (2019) and Mrini et al. (2020) used additional constituency parsing information so they are not comparable to other methods directly. Dozat and Manning (2017) introduced a biaffine model as a graph-based dependency parsing approach. Wang and Tu (2020) proposed a second-order graph-based method with message passing. Yang and Tu (2022) developed a projective parsing method based on headed spans. Lin et al. (2022) introduced a sequence generation-based parsing method, while Amini et al. (2023) leveraged structural tags and sequential tag decoding. Park et al. (2019) and Lim and Kim (2021) constructed a dependency parser using the Korean morpheme version of BERT.

Model	UAS	LAS
Park et al. (2019)	94.06	92.00
Lim and Kim (2021)	**94.76**	**92.79**
SPT-DP	94.52	92.36

Table 3: Results on Sejong dataset

5.4 Evaluation Methods

Following Wang and Tu (2020) and Amini et al. (2023), we report the Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) for PTB and UD2.2 evaluations, averaged over three random seeds and excluding all punctuation marks.

5.5 Experimental Results

Table 2 presents the performance comparison of different models on the PTB dataset. Our method achieves the SOTA performance (97.41) in UAS and competitive one (96.16) in LAS, even though it only uses **a pre-trained language model** without any additional complex modules. This highlights the strength of our **lightweight and efficient design** among top-performing models.

	bg	ca	cs	de	en	es	fr	it	nl	no	ro	ru	Avg.
base	92.63	92.90	**94.12**	**84.53**	**90.55**	92.13	**91.32**	93.63	91.69	**93.17**	88.60	**94.98**	91.63
global	**93.69**	**93.14**	93.23	84.02	90.12	**92.52**	91.20	**93.64**	**91.98**	92.97	**88.73**	94.27	91.63
unseen	81.65	88.70	80.18	77.66	73.59	88.41	83.70	83.20	78.91	79.59	77.05	79.84	81.04

Table 4: LAS scores for the UD2.2 dataset. The first row lists the languages in the test set. **base** represents the performance of a model trained individually for each language. **global** represents the performance of a single multilingual model trained on all languages. **unseen** represents the performance of individual models trained on all languages except for the test language, evaluating zero-shot transfer performance.

As shown in Table 1, XLM-RoBERTa-large, achieves SOTA results on only one language (cs). Since our approach solely relies on pre-trained language models, its performance is inherently dependent on the quality and coverage of those models. That is a reason why the relatively weak results are observed in the multilingual setting of UD 2.2, where the language model was trained across multiple languages. Nevertheless, we think that the overall performance remains reasonably strong.

In addition, we construct the **global**[4] model by training on the entire languages of UD 2.2 dataset and it leads to overall improvements in parsing accuracy across many languages. On the other hand, the results on the **Sejong** dataset (Table 3) demonstrate that our method achieves performance on par with more complex SOTA models, even though it is based on **lighter and more efficient architecture** for **agglutinative languages**.

6 Analysis

6.1 Unified Cross-lingual Dependency Parsing

Furthermore, our evaluation of language-specific experiments on the UD 2.2 dataset is expanded to cross-lingual experiments. Since the composition of dependency relation labels varies across languages, we integrate dependency relation labels from 12 languages into a shared vocabulary to construct a unified model for cross-lingual dependency parsing. We train a unified model using integrated dependency labels, referred to as the **global** model in Table 4. This cross-lingual model shows robust and competitive performance, even surpassing the **base**[5] model in several languages, as shown in Table 4. For out-of-domain evaluation, we also train an **unseen** model by excluding the target language's training data. Despite this, the **unseen** model performs well, highlighting cross-lingual correlations and the scalability of our approach. All three models use XLM-RoBERTa-large as the backbone.

6.2 Length Robustness

In previous studies, dependency parsing first attempt to represent the syntactic information of words in a sentence by feeding the final hidden states from the pre-trained model into additional modules, and classifies each embedding or directly compares the output embeddings of each word to find dependency relations. In contrast, our study newly defines and utilizes index prompts P_{idx}(P_{abs}, P_{ref}); P_{ref} in SPT of a word indirectly and they refer to the P_{abs} in SPT of another word to represent their dependency relations. Therefore, we have to check out how the relations between P_{idx} tokens are well trained by the proposed method. Table 5 presents the performance according to sentence length. Table 6 shows performance details based on the index range of templates. We observe that predictions for dependency relations involving higher indices tend to exhibit lower accuracy, which we attribute to the data distribution in the training set. To address this issue, we conducted experiments under extreme conditions to explore potential solutions (Table 7). we assumed that the training data only contains sentences with 15 or fewer words, and tested on sentences with lengths up to 40. As expected, performance decreases behind 15th position word. We attempt to extend the length of sentence by concatenating the three training sentences. After the usage of this extended training data, we obtain a significant improvement

Sentence Length Range	# of Sentences	UAS	LAS
1–10	270	97.40	96.30
11–20	764	97.70	96.36
21–30	778	97.46	96.20
31–40	433	97.39	96.20
41–50	135	97.50	96.33
51–60	28	94.55	93.48
61–70	8	98.18	97.73

Table 5: Performance statistics based on sentence length.

[4]This approach will be discussed in detail in 6.1

[5]A model trained and tested on the same language.

in performance as shown Table 7.

Index range	# of Indices	UAS	LAS
1-10	21146	97.99	96.97
11-20	15972	97.20	95.71
21-30	8580	96.72	95.51
31-40	3131	97.06	95.53
41-50	752	96.01	95.08
51-60	151	95.36	94.70
61-70	17	100.00	100.00

Table 6: Statistics and performances according to index ranges

Index Range	Before		Concatenated (after)	
	UAS	LAS	UAS	LAS
1–10	92.50	89.36	88.53 (-3.97)	79.33 (-10.03)
11–20	52.89	49.91	86.59 (+33.70)	76.20 (+26.29)
21–30	5.19	4.61	86.81 (+81.62)	76.01 (+71.40)
31–40	1.51	1.32	87.95 (+86.44)	76.85 (+75.53)

Table 7: Performance improvement after the usage of multiple sentence concatenation.

6.3 Decoder-based Models

The proposed **SPT-DP** method entirely operates at the **text level** and it enables our model to have both of the easily applicable and trainable abilities. In this section, we validate its feasibility on different language model architectures, including encoder-decoder models and decoder-only models. In these models, their decoders generate dependency parsing results of P_{ref} and $P_{\mathcal{L}}$ along with other input sequence. The results are presented in Table 8.

Decoder-based Model	UAS	LAS
XLNet-large(encoder)	97.41	96.16
T5-base	95.30	93.86
Bart-large	95.84	94.73
Llama3.2-3B	94.55	93.27
Qwen2.5-3B	94.97	93.81

Table 8: Results on PTB with Decoder-based models

6.4 Ablation Study for Prompts

First, we aim to examine the impact of each proposed prompt on parsing performance. We conduct additional experiments by excluding each of P_{abs} and P_{POS} to verify how important they are for dependency parsing. As shown in the Table 9, the role of POS information is not critical but P_{abs} has a significant impact on performance. Intuitively, when a physically referable index prompt P_{abs} exists in the input, the model can effectively refer it through transformer's attention mechanism. A detailed analysis is provided in Appendix E.

Method	UAS	LAS
SPT-DP	97.41	96.16
SPT-DP (w/o P_{abs})	96.29 (-1.13)	94.60(-1.60)
SPT-DP (w/o P_{POS})	97.25(-0.17)	95.59(-0.61)

Table 9: Effect of prompts on PTB dataset.

6.5 Inference Efficiency

Hexatagging (Amini et al., 2023), which is utilizing sequential labeling, achieves a processing speed 10 times faster than the biaffine model because it does not require additional modules, and similarly to our model, DPSG (Lin et al., 2022) also adopts a text-to-text approach on generative model. As a result, although our approach has to increase sentence length due to added prompts, it obtains faster inference speed than other conventional methods because it relies solely on a pre-trained model.

Dataset	Speed(sent/s)		
	SPT-DP (XLNet-Large)	Hexatagging (XLNet-Large)	DPSG (T5-Base)
PTB-test	39.77	28.42	-
UD 2.2 (bg-dev)	38.58	-	0.85

Table 10: Processing speed (sentences per second) of different models on PTB (test) and UD 2.2 (bg-dev)

7 Conclusions

In this paper, we introduce **SPT-DP**, as a structuralized prompt template-based dependency parsing method. Our approach enables text-to-text dependency parsing through prompt engineering by utilizing additional tokens while relying solely on pre-trained encoder models without requiring any additional modules. Despite relying solely on a pre-trained encoder model, our proposed method achieves performance higher or similar to existing models. Through experiments on the UD 2.2, we integrated dependency relation labels to develop a universal model applicable across 12 languages. This model not only enables multi-language dependency parsing within a single model but also demonstrates the ability to generalize to unseen languages to some extent. Finally, we applied our method to decoder-based models, demonstrating its applicability across different model types. Therefore, our method has several strong points; it can be easily applied to various pre-trained models appropriate for the target language or training environments, and it achieves fast inference speeds.

Limitations

In our method, there is a limitation with sequence length. Due to the large number of prompts being added, memory consumption inevitably increases. Although the model directly generates outputs without relying on additional processing pipelines—leading to faster inference—memory usage remains a concern. Moreover, as discussed in Section 6.5, there may be index positions that are not well-trained depending on the sentence length. While we have proposed a solution to this issue, it clearly remains a challenge that needs to be addressed.

Ethics Statement

We perform dependency parsing using a pre-trained model. The datasets may contain ethical issues or biased sentences, but the model does not influence them through dependency parsing.

References

Afra Amini and Ryan Cotterell. 2022. On parsing as tagging. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8884–8900, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK. Coling 2008 Organizing Committee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. 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*. OpenReview.net.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Joonho Lim and Hyunki Kim. 2021. Korean dependency parsing using token-level contextual representation in pre-trained language model. *Journal of KIISE Vol.48 No.1*, pages 27–34.

Boda Lin, Zijun Yao, Jiaxin Shi, Shulin Cao, Binghao Tang, Si Li, Yong Luo, Juanzi Li, and Lei Hou. 2022. Dependency parsing via sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7339–7353, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

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

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Joakim Nivre et al. 2018. Universal dependencies 2.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Cheoneum Park, Changki Lee, Joonho Lim, and Hyunki Kim. 2019. Korean dependency parsing with bert. In *Proc. of the KIISE Korea Computer Congress*, pages 533–536.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert Vacareanu, George Caique Gouveia Barbosa, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2020. Parsing as tagging. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5225–5231, Marseille, France. European Language Resources Association.

Xinyu Wang and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2022. Headed-span-based projective dependency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2188–2200, Dublin, Ireland. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. 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, Florence, Italy. Association for Computational Linguistics.

A Notations

B Implementation Details

For experiments for PTB, xlnet-large-cased[6] are used. For experiments for UD 2.2, bert-multilingual-cased[7], xlm-roberta-large[8] and xlnet-large-cased are used. For decoder-based models, T5-base[9], Bart-large[10], Qwen2.5-3B[11], Llama3.2-3B[12] are used. For the Korean Sejong dataset, RoBERTa-large[13], a pre-trained model for the Korean language, is used. Experiments are conducted on an NVIDIA RTX A6000. The models are fine-tuned with a batch size of 8, a learning rate of 1e-5, and 10 training epochs. Training is performed using the linear scheduler and AdamW optimizer.

C Efficiency test

For the efficiency test between Hexatagging and SPT-DP, we used the PTB test dataset to evaluate the speed of dependency parsing. For the efficiency test between DPSG and SPT-DP, we used the UD 2.2-bg dev dataset to evaluate the speed of dependency parsing. We set the batch size to 1 and conducted the experiment under the same conditions using a single A6000 GPU.

D Decoder-based Model

Unlike encoders, the decoder-based model required constrained generation. During the inference stage, contents other than P_{ref} and $P_{\mathcal{L}}$ were forcibly inserted into the sequence at intervals, allowing the model to perform accurate dependency parsing in a restricted environment.

E Analysis : Ablation Study for Prompts

To elaborate further, in experiments without P_{abs}, the order of the template implicitly replaced P_{abs}

[6] https://huggingface.co/xlnet-large-cased
[7] https://huggingface.co/bert-base-multilingual-cased
[8] https://huggingface.co/FacebookAI/xlm-roberta-large
[9] https://huggingface.co/google-t5/t5-base
[10] https://huggingface.co/facebook/bart-large
[11] https://huggingface.co/Qwen/Qwen2.5-3B
[12] https://huggingface.co/meta-llama/Llama-3.2-3B
[13] https://huggingface.co/klue/roberta-large

Notations	Components	Type	Description
w_i	-	String	Text representation of i'th word in sentence
S	w_i	String	Sentence
H_i	-	Integer	Index of the head word
$L_{(w_i,w_{H_i})}$	-	String	Dependency relation label between $w_i w_{H_i}$
r_i	$H_i, L_{(w_i,w_{H_i})}$	Set	Dependency relation of w_i
R_S	r_i	Set	Dependency relations of sentence S
$\mathcal{L}$	$L_{(w_i,w_{H_i})}$	Set	Predefined dependency relation labels
S_{dep}	w_i, r_i	Set	Sentence that include dependency parsing information
P_{abs}	"[1]","[2]",...,"[n]"	Set of strings	absolute index prompts that located at the first position of the prompt set.
P_{ref}	"[H_1]","[H_2]",...,"[H_n]"	Set of strings	reference index prompts that located in the second position of the prompt set.
P_{idx}	{P_{abs}, P_{ref}}	Set of strings	String form of index prompts, indies are encapsulated with "[]"
$P_{\mathcal{L}}$	"[acomp]", "[advcl]", ..., "[xcomp]"	Set of strings	Prompt tokens of Dependency relation labels , string of labels are encapsulated with "[]"
P_{POS}	"[NN]", "[NNP]", ..., "[WRB]"	Set of strings	Prompt tokens of pos-tags $\mathcal{P}$, string of pos-tags are encapsulated with "[]"
P_{MASK}	"[HEAD]","[DEP]"	Set of strings	Masking prompts for training
D	P_{idx},$P_{\mathcal{L}}$,P_{POS},S	String	Structuralized Prompt Template
D_M	P_{MASK},P_{POS},S	String	Masked Structuralized Prompt Template

Table 11: Notations

and was used for prediction. The experimental results (Table 9) indicate that the explicit presence of P_{abs} (physically exists), allowing for direct reference, plays a crucial role in dependency parsing through attention. In Figure 5, presents a heatmap representation of attention scores across layers and the cosine similarity of each hidden state. The attention scores on the left show that as the layers progress, the values converge, with the token in P_{ref} exhibiting high attention scores toward P_{abs} , which it is supposed to reference.

F Licenses

The PTB dataset is licensed under LDC User Agreement. The UD 2.2 dataset is licensed under the Universal Dependencies License Agreement.

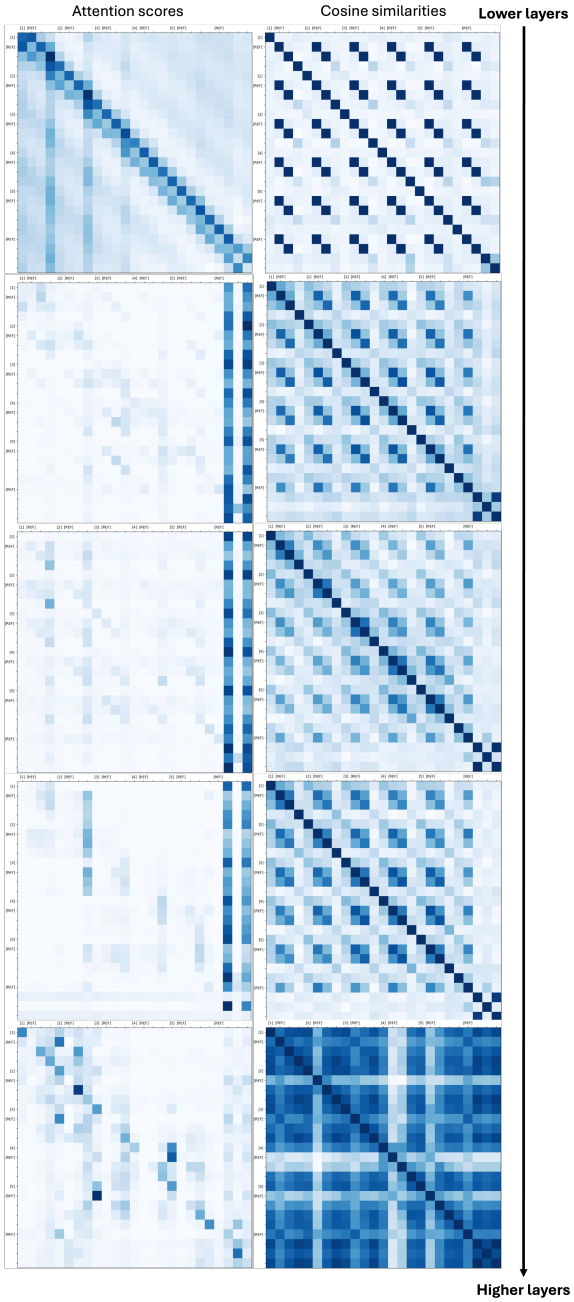


Figure 5: A heatmap of Attention scores and cosine similarities in hidden layer