Bidirectional Masked Self-attention and N-gram Span Attention for Constituency Parsing

Anonymous ACL submission

Abstract

Attention mechanisms have become a crucial aspect of deep learning, particularly in natural language processing (NLP) tasks. However, in tasks such as constituency parsing, attention mechanisms can lack the directional information needed to form sentence spans. To address this issue, we propose a **B**idirectional masked and **N**-gram span **A**ttention (BNA) model, which is designed by modifying the attention mechanisms to capture the explicit dependencies between each word and enhance the representation of the output span vectors. The proposed model achieves state-of-the-art performance on the Penn Treebank and Chinese Treebank datasets, with F1 scores of 96.47 and 94.15, respectively. Ablation studies and analysis show that our proposed BNA model effectively captures sentence structure by contextualizing each word in a sentence through bidirectional dependencies and enhancing span representation.[1]

1 Introduction

The concept of attention has become a major aspect of deep learning, and improving attention is essential to enhance the model efficacy. In natural language processing (NLP), numerous studies that utilize the sequence-to-sequence model have achieved significant performance improvements by modifying the attention mechanisms to specific tasks. Tasks such as summarization (Duan et al., 2019; Wang et al., 2018), translation (Zeng et al., 2021; Lu et al., 2021), question answering (Wang et al., 2021; Chen et al., 2019), and multi-modal learning (Nishihara et al., 2020; Liu et al., 2022) are examples of the efficacy of such mechanisms in improving model performance.

In the constituency parsing task, which involves identifying constituent phrases and their relationships in a sentence, attention mechanisms, especially self-attention, improves the performance of a parser. Many studies on constituency parsing have emphasized the importance of comprehending sentence spans to improve parser performance (Cross and Huang, 2016; Stern et al., 2017; Gaddy et al., 2018). Recent studies that incorporate attention mechanisms train parsers to comprehend sentence spans by referring to the n-grams of a sentence as the span (Tian et al., 2020) or by considering the directional and positional dependencies from splited word representation (Kitaev and Klein, 2018; Mrini et al., 2020).

However, because attention mechanisms compute the dependency of each element simultane-

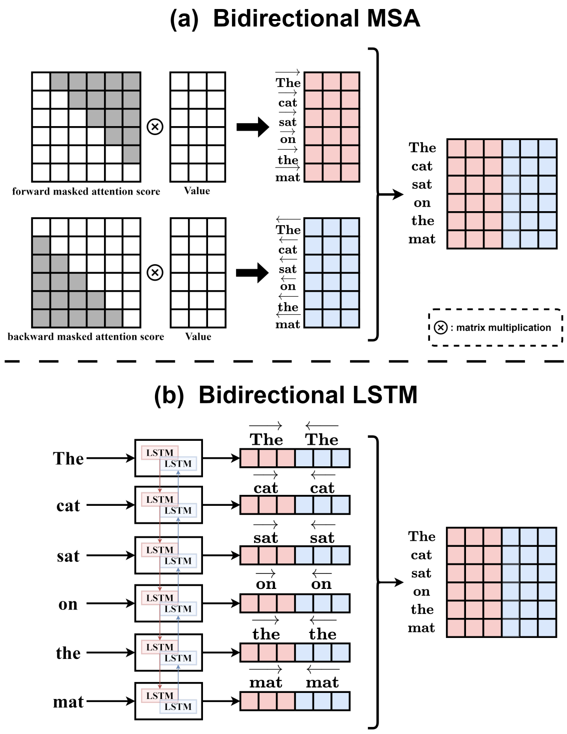

Figure 1: Comparison of the process of capturing directional information from words using BiMSA (a) and BiLSTM (b) methods in a matrix representation. In BiMSA (a), the gray area in the attention score refers to the region where directional masking has been applied.

[1] Our code is available at https://anonymous.4open.science/r/BNA-DA88.

ously, there can be a lack of the directional information that is needed to form sentence spans. This contrasts with long short-term memory (LSTM) models that consider directional information. In attention mechanisms that use attention weights between the query and key vectors as relational information between each element, the weights are computed regardless of the element's relative position. Previous studies (Kitaev and Klein, 2018; Mrini et al., 2020) acknowledged that this method could be problematic and made efforts to address it. However, such attempts were conducted under the assumption of ideal learning conditions, and the problem in the calculation process has persisted.

The purpose of this paper is to modify the attention mechanism into two types of capability. The first one obtains explicit directional information for each word, similar to the approach used by bidirectional LSTM (Figure 1(b)). The second one enhances the representation of each word by incorporating information from spans, which are suitable for constituency parsing.

In this work, we propose a novel model called **BNA** (**B**idirectional masked and **N**gram span **A**ttention). BNA employs a variant of masked self-attention (MSA) in which each element in a sequence is considered sequentially by its attention weights bidirectionally, rather than simultaneously. Moreover, BNA incorporates a novel span attention mechanism that represents a key-value matrix by subtracting the hidden states at the span boundaries. This approach enables the query (i.e., word sequence) to access the contextual information of n spans in a sentence.

Our parser achieves state-of-the-art performance with F1 scores of 96.47 and 94.15 for the Penn Treebank and Chinese Treebank datasets, respectively. In addition, through ablation study and analysis, we demonstrate that our proposed BNA model effectively captures sentence structure by contextualizing each word in a sentence through bidirectional dependencies and enhancing span representation.

2 Related Work

In the field of constituency parsing, since the introduction of the span-based approach by Stern et al. (2017), chart-based neural parsers have outperformed transition-based ones (Zhang, 2020). The span-based approach involves labeling specific text spans instead of individual tokens or words, enabling the parsers to consider the context and relationships between different spans of the sentence.

With the rise of the Transformer model (Vaswani et al., 2017) in NLP, attention mechanisms have become an attractive alternative to LSTM networks. In constituency parsing, attention mechanisms have shown promising results, as demonstrated by Kitaev and Klein (2018), who used a self-attentive network applied to the span-based parser to improve performance. They split the input vector into content and position representations and performed self-attention on each component separately. Building on this work, Mrini et al. (2020) introduced label attention layers, a modified form of self-attention that enables the model to learn label-specific views of the input sentence. In this mechanism, the attention heads are split into half, forward and backward representations, which are then used to construct span vectors of the input sentence. More recently, Tian et al. (2020) proposed span attention, which assumes no strong dependency between each hidden vector in a transformer-based encoder. Their method involves enhancing the span representation by summing the attention vector of n-grams consisting of embedded word vectors with the span vector, without using directional vectors.

However, conventional attention mechanisms treat all elements simultaneously without considering directional dependencies, making it challenging to construct span vectors using an encoder based on the attention mechanism. Furthermore, constructing arbitrary span vectors from embedded words that lack contextual information of the sentence could be improved.

In this paper, we introduce two types of attention mechanisms that address the issue of directional dependencies and that strengthen span representation.

3 Background

Self-attention is a powerful mechanism that enables neural networks to capture dependencies between different parts of a sequence. The basic idea behind self-attention is to compute a representation of the entire sequence by weighting the importance of different elements in the sequence based on their similarity to each other.

In a typical self-attention sub-layer, the sequence of input vectors $\boldsymbol{X} = [x_1, ..., x_n]$ is transformed into three sequences of vectors: queries $\boldsymbol{Q} = [q_1, ..., q_n]$, keys $\boldsymbol{K} = [k_1, ..., k_n]$, and values

$\boldsymbol{V} = [v_1, ..., v_n]$. These sequences are computed using learned linear projections:

$$\begin{aligned} \boldsymbol{q}_i &= W^Q \boldsymbol{x}_i, \\ \boldsymbol{k}_i &= W^K \boldsymbol{x}_i, \\ \boldsymbol{v}_i &= W^V \boldsymbol{x}_i, \end{aligned} \tag{1}$$

where W^Q, W^K, and W^V are learned weight matrices.

Attention weights $\alpha_{i,j}$ are computed as the dot product of the query vector $\boldsymbol{q}$ at position i and the key vector $\boldsymbol{k}$ at position j, which is subsequently normalized using the softmax function as follows:

$$\alpha_{i,j} = \text{Softmax}(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^\intercal}{\sqrt{d}}), \tag{2}$$

where d is the dimensionality of the key vectors. The $\sqrt{d}$ is used to prevent numerical instability.

Finally, the weighted sum of the value vectors is computed using the attention weights:

$$\boldsymbol{h}_i = \sum_j^n \alpha_{i,j} \boldsymbol{v}_j. \tag{3}$$

This weighted sum $\boldsymbol{h}_i$ can be seen as a hidden representation of the i-th vector that considers the importance of each of the other vectors in the sequence.

4 Approach

Our approach is motivated by the problem that self-attention mechanisms struggle to encode the relative positions and sequential order of elements within the context of a sequence (Ambartsoumian and Popowich, 2018; Hahn, 2020). Studies have been conducted to resolve this issue in tasks that require bidirectional information, such as relation extraction (Du et al., 2018) and machine translation (Bugliarello and Okazaki, 2020). To address this issue, we propose the Bidirectional Masked Self-Attention (BiMSA) and N-gram Span Attention (NSA) mechanisms. Together, these two attention mechanisms comprise our **Bi**directional masked and **N**-gram span **A**ttention (**BNA**) model.

Section 4.1 provides a brief overview of the constituency parsing process. Section 4.2 provides a more detailed explanation of BiMSA and NSA and how they are integrated into the BNA model.

4.1 Constituency Parsing

Constituency parsing is the process of analyzing the grammatical structure of a sentence by separating it down into a set of labeled spans represented by the parse tree T. The tree T of a sentence is expressed as a set of labeled spans,

$$T = \{(i_t, j_t, l_t) : t = 1, ..., |T|\}, \tag{4}$$

where the fencepost position of the t-th span is indicated by i_t and j_t, and the span has the label l_t. The parser assigns a score $s(T)$ to each parse tree T, which decomposes as

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l). \tag{5}$$

To generate the parse tree T for a given sentence $X = [x_1, x_2, ..., x_n]$, the encoder first transforms the input sequence into a set of hidden representations $H = [h_1, h_2, ..., h_n]$. Hidden vector $V_{i,j}$ for a span (i, j) is calculated as the difference between the start and end hidden vectors of that span, following the definition of Gaddy et al. (2018) and Kitaev and Klein (2018):

$$V_{i,j} = [h_j^f - h_i^f; h_i^b - h_j^b], \tag{6}$$

where h_k represents the hidden vector at position k and is constructed from two vectors from different directions, forward with h_k^f and backward with h_k^b.

The multi-layer perceptron (MLP) classifier, which serves as a decoder, takes the hidden vector $V_{i,j}$ as the input and assigns a label score to each span. The optimal parse tree

$$\hat{T} = \arg\max_T s(T) \tag{7}$$

with the highest score can be identified efficiently through a variant of the CKY algorithm.[2]

To find the correct tree T^*, the model is trained to meet the margin constraints

$$s(T^*) \geq s(T) + \Delta(T, T^*) \tag{8}$$

for all trees T through the process of minimizing the hinge loss

$$\max(0, \max_T[s(T) + \Delta(T, T^*)] - s(T^*)) \tag{9}$$

where Δ denotes the Hamming loss.

[2] We follow the parsing strategy proposed by Stern et al. (2017) and modified by Gaddy et al. (2018). For more details, see Gaddy et al. (2018)

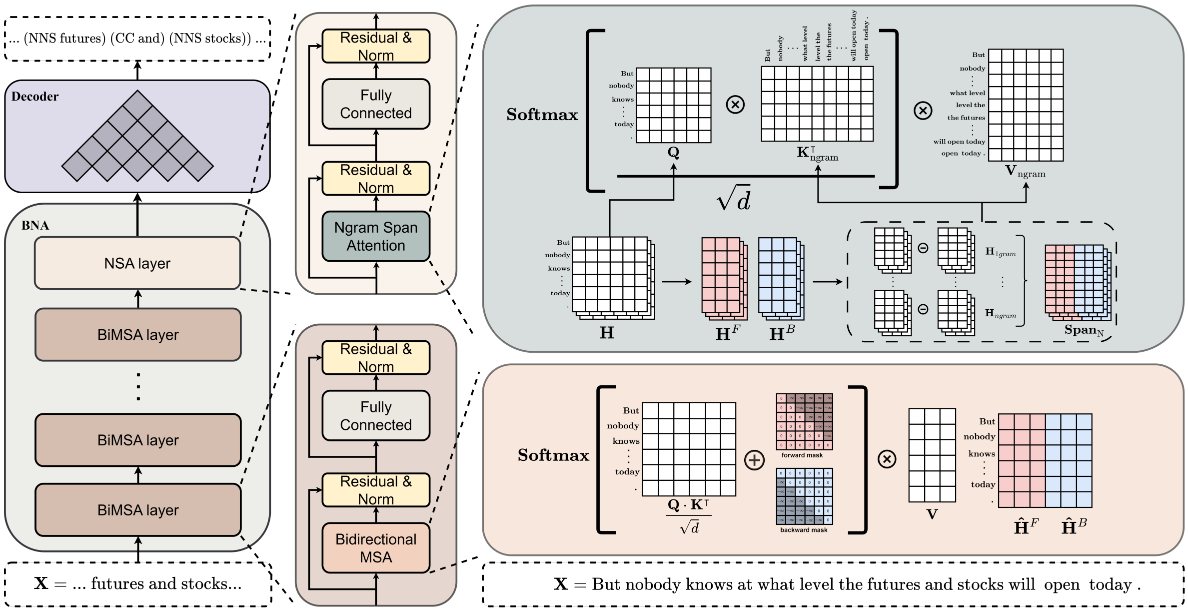

Figure 2: Our parser combines a chart decoder with an encoder, the proposed BNA model. The right side of the figure illustrates the procedure of each attention mechanism when the input sentence $\mathbf{X}$ is provided. The multiplication symbol denotes the matrix multiplication, and the summation and subtraction symbols represent the element-wise summation and subtraction, respectively.

4.2 BNA

The proposed BNA encoder is composed of two variants of the transformer encoder layers: a BiMSA layer and an NSA layer. The overall architecture of the parser is illustrated in Figure 2.

The BiMSA layer is composed of BiMSA and the position-wise feed-forward network (FFN) with the residual connection. The BiMSA layer is computed as follows:

$$\begin{aligned} \hat{\boldsymbol{H}}^{\boldsymbol{l}} &= \mathrm{LN}(\boldsymbol{H}^{\boldsymbol{l-1}} + \mathrm{BiMSA}(\boldsymbol{H}^{\boldsymbol{l-1}})), \\ \boldsymbol{H}^{\boldsymbol{l}} &= \mathrm{LN}(\hat{\boldsymbol{H}}^{\boldsymbol{l}} + \mathrm{FFN}(\hat{\boldsymbol{H}}^{\boldsymbol{l}})), \end{aligned} \tag{10}$$

where $\boldsymbol{H}^{\boldsymbol{l-1}}$ is the hidden state of the previous encoder layer and $\mathrm{LN}(\cdot)$ is the layer normalization.

The NSA layer has the same structure as the BiMSA layer, but uses NSA instead of BiMSA:

$$\begin{aligned} \hat{\boldsymbol{H}}^{\boldsymbol{l+1}} &= \mathrm{LN}(\boldsymbol{H}^{\boldsymbol{l}} + \mathrm{NSA}(\boldsymbol{H}^{\boldsymbol{l}})), \\ \boldsymbol{H}^{\boldsymbol{l+1}} &= \mathrm{LN}(\boldsymbol{H}^{\boldsymbol{l+1}} + \mathrm{FFN}(\hat{\boldsymbol{H}}^{\boldsymbol{l+1}})). \end{aligned} \tag{11}$$

Overall, BNA is composed of a sequential structure that contextualizes each word by leveraging both the sequential and directional dependencies using the BiMSA layer first and then enhances the span representation using the NSA layer.

4.2.1 Bidirectional Masked Self-Attention

BiLSTM uses forward and backward recurrent operations to produce an output vector with sequence information as the inductive bias. However, attention-based models compute attention weights solely based on the similarity between the query and key vectors and do not consider the order of elements in the sequence, making it challenging to incorporate sequence directionality.

To overcome this constraint, we introduce BiMSA to capture the directional dependency of the context, which is crucial for constructing a span vector by adding hard mask $\boldsymbol{M}$ to the scaled dot product of the query and key (Figure 1(a)). In this way, Eq. (2) is redefined as follows:

$$\alpha_{i,j} = \mathrm{Softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j^{\intercal}}{\sqrt{d}} + \boldsymbol{M}_{i,j}\right). \tag{12}$$

When $\boldsymbol{M}_{i,j}$ is equal to negative infinity, the q_i word does not affect the k_j word. Conversely, when $\boldsymbol{M}_{i,j}$ is equal to 0, it does not influence the attention weights.

The mask is divided into two distinct directional segments, namely the forward mask $\boldsymbol{M}^F$ and backward mask $\boldsymbol{M}^B$:

$$\begin{aligned} \boldsymbol{M}^F_{i,j} &= \begin{cases} 0, & i \leq j \\ -\infty, & else \end{cases} \\ \boldsymbol{M}^B_{i,j} &= \begin{cases} 0, & i \geq j \\ -\infty, & else \end{cases} \end{aligned} \tag{13}$$

We apply a forward and backward mask separately to split the directional representation of each word into its respective forward and backward components. Eq. (3) is redefined as follows:

$$\hat{\boldsymbol{h}}_i^F = \sum_j^n \alpha_{i,j}^F \boldsymbol{v}_j, \qquad \hat{\boldsymbol{h}}_i^B = \sum_j^n \alpha_{i,j}^B \boldsymbol{v}_j. \tag{14}$$

The output of BiMSA is produced by concatenating two directional hidden states into a single output representation.[3]

By using directional masks, words are constrained to attend solely to the preceding or subsequent words, enabling the model to more effectively capture the temporal dependencies. We adopt an approach of intentionally separating the bidirectional representations to construct spans from the hidden states of words. Further details are described in the following section.

4.2.2 N-gram Span Attention

The key aspect of constituency parsing is to accurately predict the contextual features of a span, represented by $V_{i,j}$. Achieving this goal requires a more fine-grained approach to modeling the contextual features.

Previous studies in constituency parsing have empirically shown that encoding spans through the subtraction of hidden states can be effective (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019; Zhou and Zhao, 2019; Mrini et al., 2020). In addition, Tian et al. (2020) recently showed that span attention can be effective for enhancing span representation. Inspired by these empirical assumptions, our novel approach NSA enables each word to reference information from various sizes of n-gram spans created from contextualized hidden states.

NSA begins by constructing an n-gram span matrix. First, the hidden states $\boldsymbol{H}$ from the previous layer are split into the forward and backward representations $\boldsymbol{H}^F$ and $\boldsymbol{H}^B$, respectively. Arbitrary span vectors are constructed by applying element-wise subtraction to the separated bidirectional hidden states, which is the same as Eq. (6):

$$\boldsymbol{H}_{ngram} = [h_j^f - h_i^f; h_i^b - h_j^b]. \tag{15}$$

The n-gram of the arbitrary span is adjusted by varying i and j.

The n-gram span matrix is constructed by concatenating the hidden states of all 1- to n-gram sequences, as follows:

$$\boldsymbol{Span}_N = [\boldsymbol{H}_{1gram}, \boldsymbol{H}_{2gram}, ..., \boldsymbol{H}_{ngram}]. \tag{16}$$

A detailed computational process for constructing the n-gram span matrix is provided in Appendix A.2.

In NSA, the query is projected from the word representation, while the key and value are projected from the span representations. The attention process enables each word to reference the contextual features from its corresponding span. Eq. (1) is redefined as:

$$\boldsymbol{Q} = W^Q \boldsymbol{H}, \quad \boldsymbol{K} = W^K \boldsymbol{Span}_N, \quad \boldsymbol{V} = W^V \boldsymbol{Span}_N. \tag{17}$$

The subsequent computations are carried out in the same manner as the self-attention process described in Section 3.

NSA allows each word to reference the contextual information from its corresponding span. It can also handle the diverse tree structures of sentences by incorporating relational information with other spans within the sentence.

5 Experiments

5.1 Datasets

To evaluate the performance of our constituency parsing model on different languages, we conduct experiments on the Penn Treebank 3 (PTB) (Marcus et al., 1993) dataset for English and the Penn Chinese Treebank 5.1 (CTB5.1) (Xue et al., 2005) dataset for Chinese.[4] We use the standard data splits for both PTB and CTB5.1.

5.2 Implementation details

To ensure a fair comparison with previous studies, we construct our model with and without the use of pre-trained models as the basic encoder. For the experiment on PTB, we utilize BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) pre-trained large models in the cased version, while for CTB5.1, we use BERT and XLNet (Cui et al.,

[3] To ensure that the output of BiMSA matches the size of the input, the dimension size of the value is set to half that of the query and key dimensions.

[4] The PTB and CTB5.1 datasets used in our experiment were officially released by the Linguistic Data Consortium. The catalog number for PTB is LDC99T42, while the catalog number for CTB5.1 is LDC2005T01.

Models	PTB			CTB5.1		
	LR	LP	F1	LR	LP	F1
Shen et al. (2018)	92.00	91.70	91.80	86.60	86.40	86.50
Teng and Zhang (2018)	92.20	92.50	92.40	86.60	88.00	87.30
Liu and Zhang (2017)	-	-	94.20	-	-	86.10
Suzuki et al. (2018)	-	-	94.32	-	-	-
Takase et al. (2018)	-	-	94.47	-	-	-
Fried et al. (2017)	-	-	94.66	-	-	-
Fried et al. (2019)	-	-	95.71	-	-	92.14
Kitaev and Klein (2018) ELMo	94.85	95.40	95.13	-	-	-
Kitaev et al. (2019) BERT	95.46	95.73	95.59	-	-	-
Kitaev et al. (2019) Ensemble	95.51	96.03	95.77	91.55	91.96	91.75
Zhou and Zhao (2019) BERT	95.70	95.98	95.84	92.03	92.33	92.18
Zhou and Zhao (2019) XLNet	96.21	96.46	96.33	-	-	-
Mrini et al. (2020) BERT/XLNet + POS	96.24	96.53	96.38	91.85	93.45	92.64
Yang and Deng (2020) BERT	95.55	96.04	95.79	93.40	93.80	93.59
Yang and Deng (2020) XLNet	96.13	96.55	96.34	-	-	-
Tian et al. (2020) BERT + POS	95.62	96.09	95.86	92.50	92.83	92.66
Tian et al. (2020) ZEN/XLNet + POS	96.19	96.61	96.40	92.42	92.61	92.52
Ours BERT	95.57	96.03	95.80	92.55	92.59	92.57
Ours BERT + POS	95.57	96.14	95.86	94.05	**94.24**	**94.15**
Ours XLNet	**96.25**	**96.69**	**96.47**	91.65	91.63	91.64
Ours XLNet + POS	96.16	96.52	96.34	**94.09**	93.83	93.96

Table 1: Comparison of labeled recall (LR), labeled precision (LP), and F1 scores of our models with those of previous studies on the test dataset. Our approach achieves state-of-the-art performance in all metrics.

2020) pre-trained base models. Following Tian et al. (2020), we use the default settings of the hyperparameters in the pre-trained models.

Kitaev and Klein (2018) experimentally demonstrated that using a character-LSTM (CharLSTM) instead of word embeddings can enhance the parsing accuracy. Therefore, to provide a fair comparison, we compare the test performance of a model that incorporates CharLSTM when a pre-trained model is not used.

In line with Kitaev and Klein (2018), Mrini et al. (2020), and Tian et al. (2020), we compare the performance of our models with and without Part-Of-Speech (POS) tagging. The POS tags are predetermined for the input sentences using the Stanford tagger (Toutanova et al., 2003). The POS tags of a given sentence are passed through the embedding layer and added element-wise to the hidden word vectors of the sentence to form the input of the model.

In our proposed NSA approach, the length of the n-gram sequence, n, should be designated as a hyperparameter. We test the performance of our model by setting n to 2, 3, 4, and 5, respectively, and select the model with the highest performance to compare it with those of previous studies. The experimental results when n is modified under the same parameter setting can be found in Section 5.5.3.

Further details on the setting of the hyperparameters for our models in all experiments are provided in Appendix A.1.

5.3 Performance comparison

The experimental results of our models and those of previous studies on the test sets are presented in Table 1. Our models outperform the previous state-of-the-art results on both datasets. Specifically, our BNA model, which does not use POS tags but employs a pre-trained XLNet model, achieves state-of-the-art performance with an F1 score 0.07 higher than the previous best score. Furthermore, the recall and precision scores show uniform improvement without bias, resulting in the highest scores among all the methods.

In the CTB5.1 dataset experiments, our models outperform the previous results by a larger margin than in the PTB experiments. Both models that use POS tags exceed the previous best performance, and the model that utilizes BERT achieves state-of-the-art performance with an F1 score improvement of 0.56.

These improved results demonstrate the effectiveness of our BNA model in resolving the critical problem of constructing span representations from the hidden states of words, which is due to the lack of dependencies between elements in attention mechanisms.

	LR	LP	F1
PTB			
Self-Attention	91.37	92.25	91.81
BiMSA	91.33	92.28	91.80
+ NSA	91.36	92.48	91.92
+ XLNet	**96.25**	**96.69**	**96.47**
+ POS	96.16	96.52	96.34
CTB5.1			
Self-Attention	83.65	85.00	84.32
BiMSA	82.44	84.67	83.54
+ NSA	83.76	85.53	84.63
+ BERT	92.55	92.59	92.57
+ POS	**94.05**	**94.24**	**94.15**

Table 2: Ablation study of the effectiveness of each approach on the test split.

	BiMSA	Self-Attn	Δ
PTB			
single model	91.80	91.81	-0.01
(+ XLNet)	96.35	96.40	-0.05
+ NSA	91.92	91.60	**0.32**
+ XLNet	96.47	96.23	**0.24**
+ POS	96.34	96.31	**0.03**
CTB5.1			
single model	83.54	84.32	-0.78
(+ BERT)	93.75	93.65	**0.10**
+ NSA	84.63	83.96	**0.67**
+ BERT	92.57	92.20	**0.37**
+ POS	94.15	94.00	**0.15**

Table 3: Comparison between the BiMSA and self-attention approaches on the test split. The row denoted by a pre-trained model in parentheses represents a case where a pre-trained model is applied to a single attention model, while Δ indicates the difference between the model performances.

5.4 Ablation study

To evaluate the effectiveness of the BiMSA and NSA modules in the BNA model, we conduct an ablation study. We compare our models with a single model of the self-attention layer, which serves as the baseline, as it is the same self-attention mechanism as the transformer encoder. For the ablation study, we start with a single model of BiMSA layers and sequentially incorporate the NSA layer, a pre-trained model, and POS tags. The hyperparameters of each model in the ablation study follow the best-performing model in Table 1.

The results presented in Table 2 demonstrate a consistent improvement in performance. Specifically, while the performance of the single model of BiMSA is comparable or inferior to that of self-attention, the inclusion of NSA leads to a performance improvement that surpasses that of the single model of self-attention. Using a pre-trained model and POS tags has been observed to be beneficial in improving performance. this finding is consistent with the results of previous studies. In particular, POS tags lead to a greater performance improvement in Chinese than in English.

Overall, it can be observed that the BiMSA and NSA models complement each other while continuously improving performance on both datasets.

5.5 Analysis

5.5.1 Directional feature for Parsing

In this section, we investigate whether the BiMSA can address the lack of directional and relative positional dependencies between words. We conduct a performance comparison between the BiMSA single model and the self-attention model, incrementally expanding the models using NSA, XLNet, and POS tags. We evaluate their performances on the test dataset using the F1 score metric. The results are presented in Table 3.

Similar to the previous ablation study results, the single BiMSA model exhibits comparable or lower performance than the single self-attention model. However, the addition of NSA significantly improves performance. This suggests that combining a model with insufficient temporal dependency and NSA may lead to a decrease in performance, but the performance enhancement in BiMSA can be attributed to the synergistic effect between BiMSA and NSA layers.

The directional and relative positional dependencies captured by the BiMSA module enable the BNA model to better handle complex syntactic structures, which is demonstrated by the higher F1 score on both the CTB5.1 and PTB datasets. This finding indicates that directional features are essential for improving parsing model performance, particularly for tasks with complex sentence structures. Moreover, the advantage of using the BNA model is even more significant for Chinese datasets, which are known for having more complex sentence structures than English.

5.5.2 Span Attention

In this section, we explore the impact of the number of NSA layers in the BNA model. Specifically, we train and evaluate models with 1, 3, 5, and 8 NSA layers, including a variant in which the order of the layers alternates between the BiMSA and NSA lay-

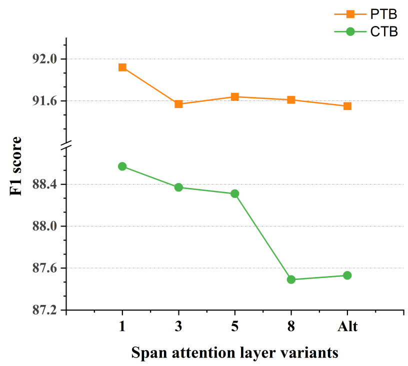

Figure 3: Comparison of the variants in NSA layers of our best-performing model and their corresponding test set F1 scores.

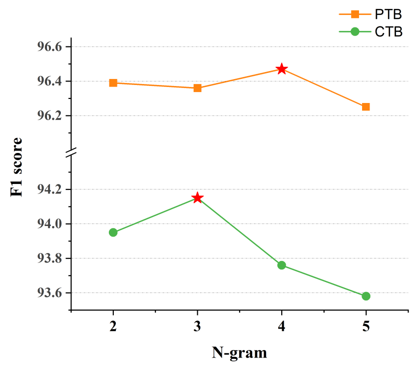

Figure 4: Comparison of the variants in the n-grams of our best-performing model and their corresponding test set F1 scores. Red stars represent our best-performing result.

ers. We maintain the total number of layers in the model as 8, and we use the same hyperparameters as those of the single model. Figure 3 illustrates the experimental results, where "Alt" refers to the alternatively applied model.

The results demonstrate that a reduced number of NSA layers leads to superior performance. This finding suggests that conducting span attention with a lack of dependency between each word in the given sentence may result in a degradation of performance. In particular, a model structure that alternates between the BiMSA and NSA layers shows no significant difference from the one that entirely consists of the NSA layer.

Overall, our experiments suggest that the selection of the number of NSA layers in the BNA model should be carefully considered, and a reduced number of layers may prove to be more effective.

5.5.3 Variations of the N-gram

To determine the optimal n-gram length for each language used in the NSA module, we conduct experiments using the best-performing BNA models in both English and Chinese. To compare the results, we vary n from 2 to 5 while keeping all hyperparameters as constant.

As shown in Figure 4, the results indicate that an n-gram length of 4 achieves the highest performance for PTB, while a 3-gram does for CTB5.1. However, extending the n-gram length beyond a certain point can lead to a decrease in model performance. As the n-gram increases, the arbitrary span becomes more similar to the given sentence. As a result, referring to a broader range of spans can dilute the span information that corresponds to each word.

However, since constituents are hierarchically composed of 2-3 words or constituents, the NSA layer allows words to refer to arbitrary spans of various positions, enabling the representation of longer spans even with a shorter span length. While it may be necessary to adjust the arbitrary span length that each word refers to depending on the language, constructing a wide range of arbitrary spans is not essential for representing sentences as constituent trees.

6 Conclusions

The primary goal of this study was to design attention mechanisms to capture the explicit dependencies between each word and enhance the representation of the output span vectors. Through our experiments, we demonstrated that our proposed BiMSA more effectively contextualizes each word in a sentence by considering the bidirectional dependencies, while NSA improves the span representation by attending to arbitrary n-gram spans. Our findings have major implications for span-based approaches in constituency parsing tasks. Specifically, applying the span representation method to the attention mechanism leads to a significant performance improvement.

In conclusion, constructing a span representation from words contextualized within a given sentence can lead to additional improvement in parsing. Overall, our study contributes to the advancement of attention mechanisms in NLP. We hope that our findings will inspire further research in this area.

Limitations

However, the weight of the model remains a significant issue for high-performance inference, especially for preprocessors that deconstruct and analyze the sentence structure before understanding it. Using a costly parser in real-time machine learning tasks can present limitations as rapid data processing is a crucial objective in this current area of research. To address this concern, future studies should focus on developing a lightweight span attention module that considers the bidirectional dependencies.

Although the n-gram span attention operation can be robust for trees of various sizes and structures, it involves concatenating n-grams from 1 to n to create an n-gram span matrix, making it a heavy operation. This limitation becomes increasingly evident as sentences become longer, resulting in a discrepancy in learning speed when compared to existing parsers during comparative experiments. Tian et al. (2020) suggested categorizing extracted n-grams in a span (i, j) by their length so that n-grams in different categories are weighted separately instead of using all n-grams. It may be helpful to modify the attention to focus only on a limited range of spans to improve the speed of the n-gram span attention module. This modification remains as future work.

References

Artaches Ambartsoumian and Fred Popowich. 2018. Self-attention: A better building block for sentiment analysis neural network classifiers. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 130–139.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing machine translation with dependency-aware self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. 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 2913–2923.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668.

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.

Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. 2018. Multi-level structured self-attentions for distantly supervised relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2216–2225.

Xiangyu Duan, Hongfei Yu, Mingming Yin, Min Zhang, Weihua Luo, and Yue Zhang. 2019. Contrastive attention mechanism for abstractive sentence summarization. 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 3044–3053.

Daniel Fried, Nikita Kitaev, and Dan Klein. 2019. Cross-domain generalization of neural constituency parsers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 323–330.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 2676–2686.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Zichen Liu, Xuyuan Liu, Yanlong Wen, Guoqing Zhao, Fen Xia, and Xiaojie Yuan. 2022. Treeman: Tree-enhanced multimodal attention network for icd coding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3054–3063.

Yu Lu, Jiali Zeng, Jiajun Zhang, Shuangzhi Wu, and Mu Li. 2021. Attention calibration for transformer in neural machine translation. 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 1288–1298.

Mitch 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 Ndapandula Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742.

Tetsuro Nishihara, Akihiro Tamura, Takashi Ninomiya, Yutaro Omote, and Hideki Nakayama. 2020. Supervised visual attention for multimodal neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4304–4314.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827.

Jun Suzuki, Sho Takase, Hidetaka Kamigaito, Makoto Morishita, and Masaaki Nagata. 2018. An empirical study of building a strong baseline for constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 612–618.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4599–4609.

Zhiyang Teng and Yue Zhang. 2018. Two local models for neural constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 119–132, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, Fei Xia, and Tong Zhang. 2020. Improving constituency parsing with span attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1691–1703.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. 2021. Cluster-former: Clustering-based sparse transformer for question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3958–3968.

Yongzhen Wang, Xiaozhong Liu, and Zheng Gao. 2018. Neural related work summarization with a joint context-driven attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1776–1786.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. *Advances in Neural Information Processing Systems*, 33:21687–21698.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Jiali Zeng, Shuangzhi Wu, Yongjing Yin, Yufan Jiang, and Mu Li. 2021. Recurrent attention for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3216–3225.

MeiShan Zhang. 2020. A survey of syntactic-semantic parsing based on constituent and dependency structures. *Science China Technological Sciences*, 63(10):1898–1920.

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.

A Appendix

A.1 Further implementation details

We employ a grid search to identify the optimal parameter settings for our model with a random seed fixed at 42. The parameter tuning was conducted across various ranges, including learning rates of 1e-5, 2e-5, and 3e-5, batch sizes of 50, 100, and 200, n-gram values of 1, 2, 3, and 4, and dropout ratios of 0.1 and 0.2 on the development set.

In the PTB dataset experiments, the optimal model achieves the highest performance with a learning rate of 2e-5, a batch size of 200, and an n-gram value of 4 for the NSA layer. The dropout ratios for the residual connections, feed-forward layer, attention, and CharLSTM morphological representations were 0.2, 0.2, 0.2, and 0.1, respectively.

In the CTB5.1 dataset experiments, the most successful model uses a learning rate of 3e-5, a batch size of 50, and an n-gram value of 3 for the NSA layer. The dropout ratios for the residual connections, feed-forward layer, attention, and CharLSTM morphological representations were 0.1, 0.1, 0.1, and 0.2, respectively.

Both experiments employed identical model sizes, with a model dimensionality of 512 and a feed-forward layer size of 1024. The query/key/value sizes were set to 64, except in the BiMSA layer, where the value size was halved to 32 for split forward and backward computations.

When the parser utilizes a pre-trained model, the number of layers is set to 2. In contrast, when a single model is employed without a pre-trained model, the architecture employs 8 layers. Additionally, to enhance the training speed and performance of the single model, a batch size of 250 and a learning rate of 0.0008 are employed.

All parsers, including those utilizing pre-trained models, were trained within a 12 hour. Training was conducted using a single NVIDIA RTX A5000 GPU for each parser. The parser without a pre-trained model has 15.9 million parameters, while the parser with a pre-trained model, which has 2 layers, has 4.7 million parameters.

A.2 Procedure of constructing arbitrary span matrix

The separated bidirectional word representations, namely $\boldsymbol{H}^F$ and $\boldsymbol{H}^B$, construct span matrices ranging from 1-gram to n-gram. These completed span matrices, $\boldsymbol{Span}_N^F$ and $\boldsymbol{Span}_N^B$, are concatenated to form a single $\boldsymbol{Span}_N$. The specific computation procedure for constructing an arbitrary n-gram span matrix with bidirectional word features is presented in Figure 5.

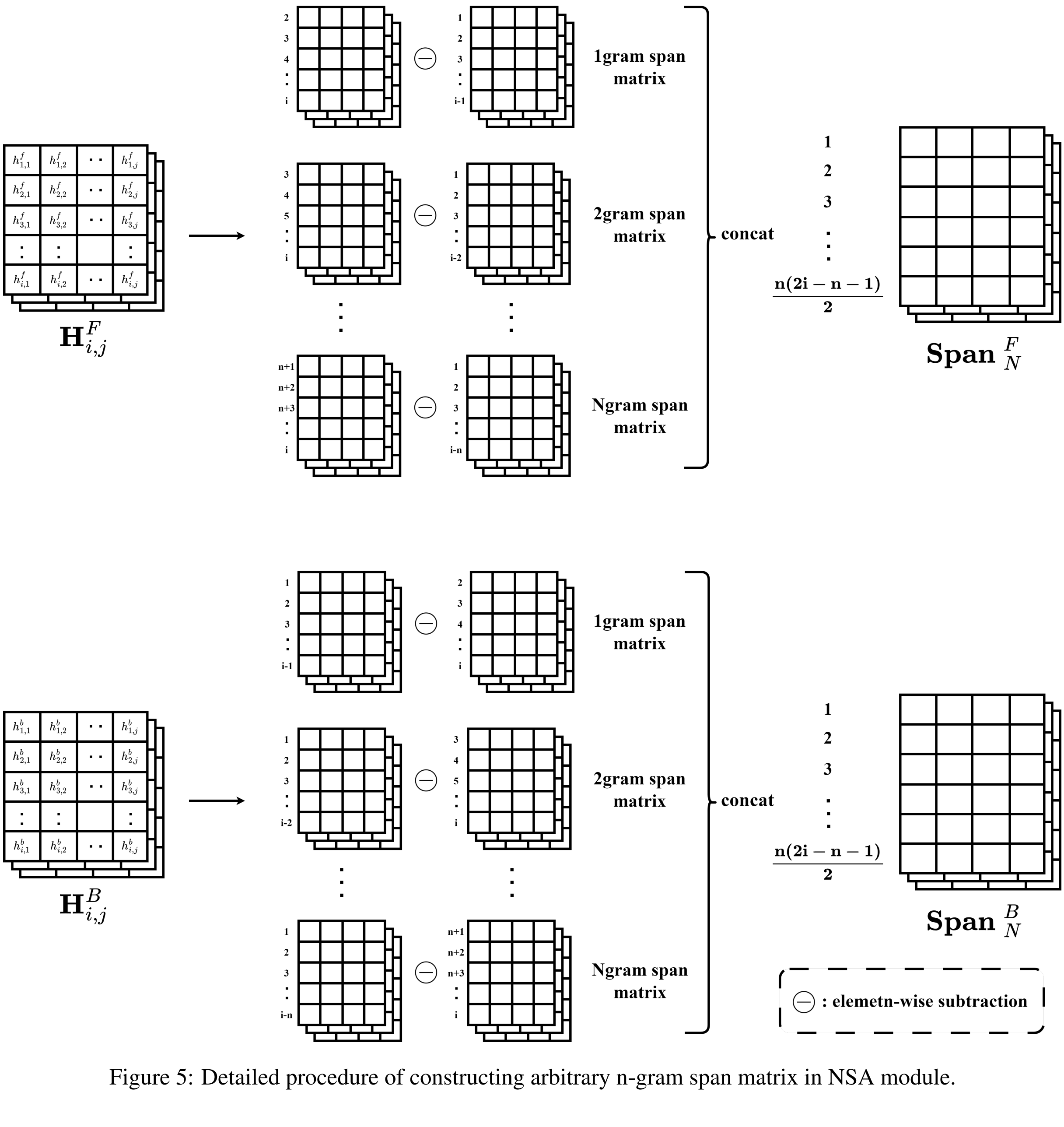

Figure 5: Detailed procedure of constructing arbitrary n-gram span matrix in NSA module.