2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Learning to Adaptively Incorporate External Syntax through Gated Self-Attention

Shengyuan Hou[1][0000-0001-7031-8688]

[1] Shanghai Jiao Tong University, Shanghai Minhang Dongchuan Road 800, China
hsyhwjsr@sjtu.edu.cn

Abstract. Transformers are known to be able to implicitly learn syntax from data during training, albeit their quality depends heavily on the data size and quality. However, introducing structured syntactic information into the sequential Transformer is not trivial. Previous analytical studies have shown that Transformers learn more abstract representations through layers, with their lower layers being more related to syntactic information. In accordance, to provide extra flexibility and interpretability along with the utilization of constituency syntax, we propose an architecture that allows different layers of the Transformer to control the incorporating weights of external syntax adaptively through a gating mechanism. Experimental results of our learned syntactic gating weights reveal that Transformer tends to utilize constituency syntax hierarchically, which nicely aligns with previous findings, showcasing the interpretability of our architecture. Moreover, experimental results on five machine translation datasets across various language pairs also show that our model outperforms the vanilla Transformer by 1.12 BLEU score on average, and it is competitive against other latest syntax-aware models. Also, only a few additional hyperparameters are required, alleviating the burden of searching for the best syntax incorporation location.

Keywords: Gating Mechanism, Constituency Syntax-aware Architecture, Machine Translation.

1 Introduction

Previous studies [26, 11, 5, 3, 9] have shown that Transformer [27] inherently learns syntax from training data in its lower layers. However, the effects rely heavily on the size and quality of the training data. Rather than relying solely on training data, integrating syntactic information from external parsers is advantageous, particularly for small- to medium-scale datasets. Many studies [25, 33, 31, 2, 15, 32, 28, 30, 10, 23, 12, 7, 16, 29, 6] have thus been proposed. For dependency syntax, some methods [15, 30] mask attention scores of tokens that are too distant in the syntax tree. Others [25, 33, 2] utilize parental or ancestral relationships to constrain each token's attention range. For constituency syntax, some methods [31] introduce additional encoder to induce the input syntax structure, and others [16] propose to fuse it into the positional embedding. By far, many latest works [12] exploit both syntactic structures, while others [29, 6] extend the induction of syntactic structure to more general natural language tasks.

More recently, some studies [15, 8] have found that Transformers are sensitive to the location when introducing syntactic structures. The constituency syntax could enhance the Transformer model substantially in lower layers but instead hinder its performance in deeper layers [8]. In contrast, dependency syntax tends to be uniformly utilized in all layers [33, 2]. Therefore, it is crucial to determine which layer or attention head the syntactic information should be incorporated into. However, most existing methods integrate syntactic structures in a hard manner, lacking interpretability since the way that Transformer utilizes syntactic information remains unclear. Worse still, the 0-1 hard incorporation entails less flexibility while enduring a higher hyperparameter searching burden of exponential complexity. Given L layers each with h attention heads, there are totally $(h+1)^L$ possible choices for hard incorporating locations.

To address the deficiencies mentioned above, in this work, we adopt syntax-aware attention proposed previously [8], namely syntactic local range (SLR), to integrate constituency syntax into Transformer. Then we develop a gating mechanism that enables the architecture to self-adaptively learn the amount of syntactic guidance for each attention head. The gating mechanism alleviates extensive hyperparameter searches for the best syntactic incorporating locations and brings extra performance gain due to better distribution of syntactic guidance through layers and attention heads. By integrating syntactic structure adaptively with soft weights, our method also entails larger flexibility and more interpretability. Furthermore, to alleviate the self-reinforcing problem [4, 1, 21] caused by the gating network during training, we also design several regularization methods to ensure the stability of gating weights.

Compared with previous fundamental work [8], our key contributions are:

- We propose a novel **adaptive** syntax-aware architecture that leverages a gating mechanism instead of hard incorporation to dynamically control the integration of external syntactic information across different layers and attention heads of the Transformer, thereby enhancing both flexibility and interpretability while reducing the requirement for extensive hyperparameter searches to identify the best syntactic integration locations.
- We validate the effectiveness of our approach on multiple machine translation datasets. Our model consistently outperforms the vanilla Transformer and other syntax-aware models. Most importantly, further analysis on learned gate values show that the Transformer tends to utilize constituency syntax mostly in lower layers and progressively alters to its own self-attention in deeper layers, which aligns nicely with previous findings on the location sensitivity of constituency syntax, validating our model's interpretability. Moreover, only few additional hyperparameter searches for the gating network are required, which can be accomplished in linear complexity.

2 Related Work

2.1 Syntactic Attention Mask

Compared with global attention, the syntactic attention mask could prohibit the attention from overweighting syntactically distant tokens over close ones. However,

inherently constituency and dependency syntax are both tree structures, which makes it hard to generate a tensorized attention mask. Previously, many methods [25, 33, 31, 2, 15, 28, 30, 23, 12, 7, 8] try to construct syntax-aware local attention patterns for not only machine translation but also other language understanding and generation tasks [29, 6].

For dependency syntax, parent-child relationships and distance among nodes in the syntax tree are frequently considered. SLA [15] restricts each token to only attend to its neighborhoods whose distance is below some threshold. PASCAL [2] builds up a Gaussian weight distribution, which is centered on the parent token's position, in each attention matrix row. SG-Net [33] restricts each token to attend to only all of its ancestral nodes. SEPREM [30] filters out distant token pairs based on the uni-directional distance from the head node to the dependent node in the syntax tree. For constituency syntax, the height of two nodes' smallest common subtree or tree slices at some depth are often considered. ST-NMT [31] constructs a sentence template by keeping the node in some fixed depth of the tree and incorporate another encoder to learn how to predict it. Distance-Transformer [8] builds up an attention mask based on the syntactic local range defined by the constituency syntactic distance. Some methods [23, 12] utilize both syntactic structures. Structformer [23] proposes to utilize both the syntactic distance generated from the constituency tree and syntactic heights generated from the dependency tree to vectorize syntactic structures. Inspired by this, Syntaxformer [12] learns syntactic vector structure unsupervisedly, where the syntactic induction and incorporation are fused during training. Although these diversified syntax-aware architectures enhance the language tasks, few of them explore how to incorporate syntactic structures adaptively, constraining their flexibility and interpretability.

2.2 Syntactic Attention Incorporation

There are three common manners to utilize a syntactic attention mask. LISA [25] takes the syntactic attention mask as a label and introduces the squared L2 distance between the label and computed attention as an auxiliary loss. ST-NMT [31] utilizes the generated template sequence as a label and then introduces a new Transformer encoder to learn it. Other latest methods [33, 2, 15, 28, 30, 12, 7] perform element-wise operations, or weighted linear combinations between the mask and attention weights. Unfortunately, most of them rely on grid searches or heuristics for where to incorporate syntax, suffering from extensive hyperparameter tuning and lacking flexibility.

3 Preliminaries

3.1 Syntactic Distance

Syntactic Distance is firstly proposed in [22] which can be used to represent the constituency tree as a list of distance values. Given a sentence $S = [t_1, t_2, \ldots, t_n]$ with length n, the syntactic distances of S is notated as $D = [d_1, d_2, \ldots, d_{n-1}]$, in which d_i is the height h_i of the sub-tree rooted by token t_i and t_{i+1}'s least common ancestor (LCA). In practice, we set the final distance to be $h_i - 1$ so that the smallest distance

is 1 and syntactic distance is a list of integers ranging from 1 to $h_T - 1$, where h_T is the height of the tree. This is optional since the downstream calculation depends on the ranking of syntactic distances rather than their values. A larger distance indicates a more distant relationship between neighborhood constituents.

For example, in Fig. 1(b), the LCA of 'I' and 'have' is node 'S', whose height is 4 in the syntax tree, so the first syntactic distance is 4 − 1 = 3, and the latter ones can be derived likewise. Accordingly, node 'S' has children 'I', 'VP', and'.', where 'VP' is a subtree in which the maximum distance is 2, so the distance between 'I' and 'VP' as well as 'VP' and '.' is 2 + 1 = 3.

3.2 Syntactic Local Range

The **syntactic local range (SLR)** [8] defines the influence of a given node on other tokens in a sentence based on the constituency tree, which can be well introduced in deep learning structures as syntactic guidance [13]. Given a syntax tree T, the SLR of one leaf node t can be decomposed into two directions: left-range and right-range, which are called pre-text and post-text SLR. Given one token t and its parent node t_p, the pre-text local range meets:

- If t is not the leftmost child of t_p, then t's SLR on the pre-text direction starts at t_p's leftmost child, and stops at t.
- If t is the leftmost child of t_p, back-trace its ancestors to find a nearest constituent v where v is not the leftmost child of its parent v_p. Then t's SLR starts at v_p.'s left-most child and stops at t.

On post-direction, SLR can be computed similarly. In Section 4, we leverage a fast algorithm utilizing syntactic distance to compute the SLR, which is equivalent to the definition mentioned above. Detailed illustrations and proof can be found in [8].

4 Methods

The overall architecture is shown in Fig. 1. In particular, we firstly parse the input using an external parser. Then the syntactic distance is calculated from the constituency tree and utilized to construct a syntactic attention mask with soft SLR proposed by [8]. The mask is then multiplied element-wisely with the Transformer's original attention (named as global attention since it doesn't constrain attention ranges) to get syntactic attention. Finally, we linearly combine the syntactic and global attention with a gating network.

4.1 Syntactic Attention

Given the **hard SLR** illustrated in section 3.2, a method based on the syntactic distance could be utilized to effectively calculate it. Then the hard SLR evolves into the **soft SLR** [8], which performs well in multi-head self-attention.

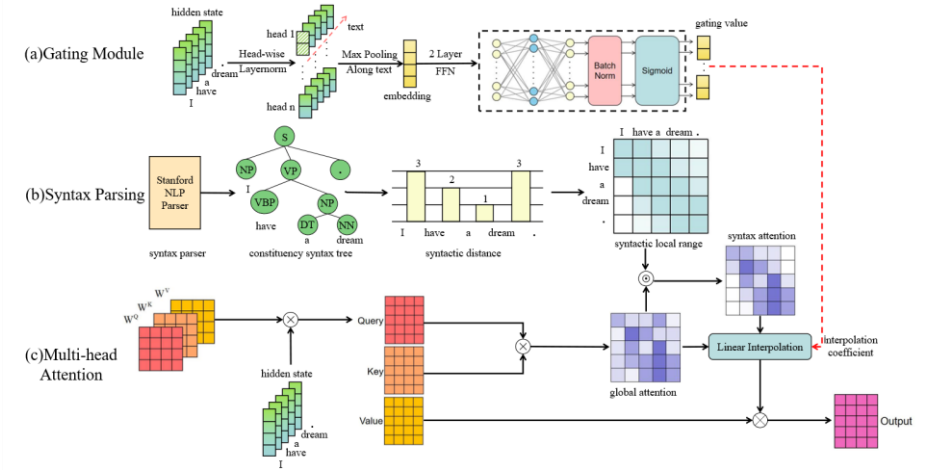


Fig. 1. Overview of gating-distance Transformer. (a) Each layer entails a gating module, which takes current hidden states as input, then performs max-pooling along text to integrate representations, next pass it to a 2-layer FFN with normalization, and finally use sigmoid to generate h gating signals (h is head number). (b) Given constituency tree, syntactic distance is pre-computed and is utilized to calculate attention mask with soft syntactic local range. (c) Syntax attention is calculated at first and combined linearly with global attention by the weights from gating modules.

Generating SLR from Syntactic Distance The gap between SLR and syntactic distance can be bridged by an algorithm proposed previously [8] to compute SLR effectively using only the syntactic distance of a given sentence. The process of generating the SLR matrix **M**, where each line of **M** represents the SLR of a token, could be formulated by introducing α_i^j as syntactic distance variance. Equation (1) defines how [8] calculates the pairwise ranking relationship of the syntactic distance by matrix operations at the first step.

$$\alpha_i^j = \mathrm{I}(d_i - d_j) \tag{1}$$

In Equation (1), $\mathrm{I}(\cdot)$ is the indicator function, with $\mathrm{I}(x) = 1$ if x is positive, else $\mathrm{I}(x) = 0$. Apparently as $d_i > d_j$, α_i^j becomes 1 and alters to 0 conversely. Then **M** can be generated by Equation (2). This makes the computation of the SLR matrix highly parallel.

$$m_{ij} = \begin{cases} \prod_{j \le t \le i-1} \alpha_t^{i-1} & j < i-1 \\ \prod_{i+1 \le t \le j} \alpha_t^{i+1} & j > i+1 \\ 1 & o.w. \end{cases} \tag{2}$$

Soft SLR In terms of soft SLR, it uses a smoothed difference of syntactic distance d_i and d_j in Equation (3), controlled by a temperature parameter τ. The $f(x) = (\tanh(x) + 1)/2$ is a smoothed version of indicator function $\mathrm{I}(\cdot)$.

$$\alpha_i^j = \frac{\tanh\left((d_i - d_j)/\tau\right)+1}{2} \tag{3}$$

Equation (3) yields 0 when $d_i \ll d_j$ and gradually increases to 1 as d_i increases. It is a soft version of (1), with more flexibility. The SLR matrix can be calculated using (2) as well. To initiate the continued cumulative multiplications, mask values on the two secondary diagonals in **M** are set to 1, as shown in (2).

4.2 Gating Structure for Self-adaptive Weight

Initially, the gating network receives the hidden states of the current layer and applies l-dim max-pooling along the sequence to aggregate multiple representations. Equation (4) formulates this mathematically and notates the input hidden state of l-th layer as $\mathbf{H}^l = [\mathbf{H}_1^l, \dots, \mathbf{H}_h^l]$, $\mathbf{H}_i^l \in R^{B\times n\times d}$, while h is the number of heads, n is the sequence length, B is the batch size and d is the hidden dimension.

$$\mathbf{T}^l = Maxpool1D(\mathbf{H}^l),\ \ \mathbf{T}^l \in R^{B\times d} \tag{4}$$

Then the aggregated hidden states $\mathbf{T}^l$ are further fed into a 2-layer feed-forward network in (5), where $\mathbf{W}_1^l \in R^{d\times d'}$, $\mathbf{b}_1^l \in R^{d'}$, $\mathbf{W}_2^l \in R^{d'\times h}$, $\mathbf{b}_2^l \in R^h$, $\mathbf{O}^l \in R^{d'\times h}$, and 'LayerNorm' refers to layer normalization. We set a gating weight for each attention head, so the output dimension is h.

$$\mathbf{O}^l = \mathbf{W}_2^l * LayerNorm\left(ReLU(\mathbf{W}_1^l * \mathbf{T}^l + \mathbf{b}_1^l)\right) + \mathbf{b}_2^l \tag{5}$$

Finally, the output is passed to the sigmoid function to generate gating signals, as shown in (6). Before the output is fed into the logistic layer, the batch normalization ('BatchNorm') is performed to ensure distributional stability.

$$\mathbf{G}^l = \sigma\big(BatchNorm(\mathbf{O}^l)\big) = [\mathbf{G}_1^l, \dots, \mathbf{G}_h^l] \tag{6}$$

4.3 Self-adaptive Incorporation of Syntactic Information

With the syntactic attention mask **M** calculated by (4), a syntax-guided multi-head attention can be generated by (7), (8), and (9).

$$\mathbf{A}_{raw} = Attn(\mathbf{Q}, \mathbf{K}) = Softmax(\frac{\mathbf{QK^T}}{\sqrt{d_k}}) \tag{7}$$

$$\mathbf{A}_{syn} = SynAttn(\mathbf{Q}, \mathbf{K}, \mathbf{M}) = \mathbf{S}_m(\mathbf{M}, \frac{\mathbf{QK^T}}{\sqrt{d_k}}) \tag{8}$$

$$[S_m(\mathbf{M}, \mathbf{X})]_{j,i} = \frac{m_{ji}e^{x_{ji}}}{\sum_{k=1}^{n} m_{jk}e^{x_{jk}}} \tag{9}$$

In (9), $S_m(\cdot)$ represents the masked softmax, while m_{ji} and x_{ji} are the elements in the j-th row and i-th column of the **M** and **X** matrices. $[\cdot]_j = [1, \dots, n]$ corresponds vertically stacking the elements yielded by the function inside it. Because of (7) and (8),

there are two attention score matrices generated totally. One is the raw attention score matrix $\mathbf{A}_{raw} \in R^{n\times n}$ without any inductive bias information, and the other is $\mathbf{A}_{syn} \in R^{n\times n}$ enhanced by syntactic relations.

Given the gating weight $\mathbf{G}_i^l$ for attention head i on layer l, we linearly combine the syntactic attention $\mathbf{A}_{syn,i}^l$ and global attention $\mathbf{A}_{raw,i}^l$ to get the final attention $\mathbf{A}_i^l \in R^{n/h\times d}$, and the final output $\mathbf{F}^l \in R^{n\times d}$ is calculated with the attention $\mathbf{A}^l \in R^{n\times n}$ and value $\mathbf{V}^l \in R^{n\times d}$, formulated in (10) and (11).

$$\mathbf{A}_i^l = \mathbf{G}_i^l * \mathbf{A}_{syn,i}^l + \left(1 - \mathbf{G}_i^l\right) * \mathbf{A}_{raw,i}^l \tag{10}$$

$$\mathbf{F}^l = \mathbf{A}^l\mathbf{V}^l = [\mathbf{A}_1^l, \dots, \mathbf{A}_n^l]\mathbf{V}^l \tag{11}$$

4.4 Over-reliance of Gating Value

A naive gating network encounters severe over-reliance towards syntactic attention during training, as shown in Fig. 3(a). We think this is caused by the self-reinforcing problem [4, 1, 21]. During the training of two experts (SLR and global), since SLR restricts the attention range, it eliminates token connections and helps model converge faster so that it is trained more rapidly and thus selected more by the gating network [21], forming a self-reinforcing process.

4.5 Regularization for Gating Structure

To tackle the problem of over-reliance on syntactic local attention, we propose several simple yet effective regularization tricks, including three components.

Pre-warmup Training The syntactic attention is mathematically formulated from the initiation of training, whereas the global attention is still random. Therefore, we want to let them collaborate initially and compete against each other after the global attention has converged better. Specifically, in the first K epochs, the gating network is frozen, yielding random weights around 0.5. After K epochs, we unfreeze these weights to let two types of attention compete. Subsequent paragraphs, however, are indented.

Batch Normalization Before outputting gate values, we incorporate a batch normalization layer (6), whose function is like DMoE [4] that adds a soft constraint for each gate to prevent extreme weights (0 or 1) from dominating.

Syntax Ignoring Previously, some studies explored incorporating random masking, similar to Dropout [24], into syntactic mask to improve generalization since the external parser could yield incorrect or ambiguous results, such as parent ignoring [2], which randomly assigns some rows of the attention matrix to 0. Here we apply vanilla Dropout on the syntactic attention. It could help the gating network output weights for syntactic attention less aggressively. If there is not such enhancement to introduce syntactic

attention somewhere, a few destructions on the syntactic attention will motivate the gating network to drop it.

5 Experiments

In section 5.1, we discuss our experimental setup. In section 5.2, we compare the results of our model with other dependency-based or constituency-based methods and the vanilla Transformer. In section 5.3, we analyze the location sensitivity of different syntax, discussing the interpretability of our gating network, as well as the impact of τ. In section 5.4, we conduct some ablation studies.

5.1 Experimental Setup

Datasets and Preprocessing We conduct experiments on 5 machine translation datasets: IWSLT14-De-En, IWSLT14-En-De, NC11-De-En, NC11-En-De, and WMT18-En-Tr. Their train/validation/test sizes are listed in Table 1. We use Moses [14] for uncased word tokenization and Sub-word NMT [20] for BPE. The Stanford CoreNLP parser [17] is used to parse the constituency tree.

IWSLT14 stems from Ted Talks with short text lengths. For preprocessing, we do uncased word tokenization and then clean up the dataset by only keeping the pairs with source-target length ratio within 1.5 and total length no longer than 175. Next, we perform BPE subword tokenization with a shared dictionary of size 10k. Finally, we randomly select 5% of training data as the validation set, and merge dev2010, dev2012, tst2010, tst2011, and tst2012 into the test set.

NC11, with medium text lengths, comes from news comments. For preprocessing, we again perform uncased word tokenization. Then we clean up the dataset by only keeping the pairs with source-target length ratio within 1 and total length within 80. Next, we use the Subword NMT toolkit for BPE tokenization with a shared dictionary of size 16k. The validation and test set are newstest2015 and newstest2016 respectively.

For WMT18-En-Tr, after uncased word segmentation, we keep sentence pairs with a length of no more than 80 and a length ratio within 1. We continue to use the Subword NMT for BPE segmentation, and the dictionary size is 16K. WMT18-En-Tr dataset is used as the training set, newstest2016 is used as the validation set, and newstest2017 is used as the test set.

Table 1. Train/validation/test sizes of different experimental datasets.

Datasets	Train	Valid	Test
IWSLT14-De/En	160239	7283	6750
NC11-En/De	238843	2169	2999
WMT18-En-Tr	207373	3000	3007

Hyperparameter We set the dimension for both encoder and decoder embeddings to 512, and the dimensions of hidden layers in both the encoder and decoder's FFN are set to 1024. Input and output sequences share the same embeddings. The number of both encoder and decoder layers is set to 6, each with 4 attention heads. For pre-warmup training, since batch normalization is entailed before gating output, all the gating values come close to 0.5 before the gating module is unlocked. We apply dropout with a ratio of 0.3 within the FFN layer, and we also apply attention dropout with a ratio of 0.2. As for optimization, we adopt the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$. We increase the learning rate from 10^{-3} linearly until it reaches the initial learning rate of 7×10^{-3} for the first 4000 training updates. After the initial learning rate is reached, we decrease it proportionally to the inverse square root of the update number. We use label smoothing of value $\epsilon_{ls} = 0.1$ and weight decay of 10^{-4}. All implementations are based on the fairseq[1] [18]. Most configurations are stemmed from previous work [8], except for the learning rate which is tuned by exponential searches. For two gating-related hyperparameters, namely, locking epoch K and ignore rate p, that are tuned by the greedy search, our configurations are shown in Table 2.

Table 2. Configurations for regularization.

Datasets	Ignore Rate	Locking Epoch
IWSLT-De-En	0.4	80
IWSLT-En-De	0	50
NC11-De-En	0.2	60
NC11-En-De	0.2	80
WMT18-En-Tr	0.1	70

Evaluation The case-sensitive 4-gram BLEU score was calculated using Sacre-BLEU[2] [19] as the evaluation metric, with a batch size of 128. For NC11-En-De, beam size and length penalty are respectively set to 4 and 0.6, while for other datasets, these two values are 5 and 1. For evaluation, we select the checkpoint with the smallest validation loss during inference. All results are averaged over 5 random seeds, and the improvements have statistical significance ($p < 0.01$).

5.2 Results

For comparison, we re-implement the following precedent works.

- LISA [25] constructs a 0-1 attention mask label where in the i-th row only the column of token i's dependency parent has value 1. It introduces an auxiliary L2 loss to enforce one attention head to learn its syntactic pattern.

[1] https://github.com/facebookresearch/fairseq
[2] https://pypi.org/project/sacrebleu/

- PASCAL [2] constructs a syntactic attention mask where each mask value $m_{i,j}$ is a probabilistic weight of Gaussian distribution centered by token i's parent position. Each token's position is fed into the Gaussian p.d.f. function.
- SLA [15] prevents each token from attending excessively distant token nodes out of a distance boundary. The distance between two nodes is defined as their shortest path length in the dependency tree. Additionally, SLA designs a naive gating network controlling the combination of syntax and global attention at each attention head of each layer. Without any regularization, its gating weights are mostly 0.5, which is meaningless.

Detailed comparisons of the translation performance on five popular machine translation tasks are shown in Table 3. These results demonstrate the superior performance of our model compared to the vanilla Transformer and other syntax-aware models in terms of BLEU score. For instance, our model achieved a 1.12 BLEU score improvement on average over the vanilla Transformer. For NC11-De-En and NC11-En-De datasets, compared with the best syntax-aware benchmarks, our model wins 0.75 and 0.53 BLEU scores respectively, highlighting the effectiveness of our adaptive gating mechanism.

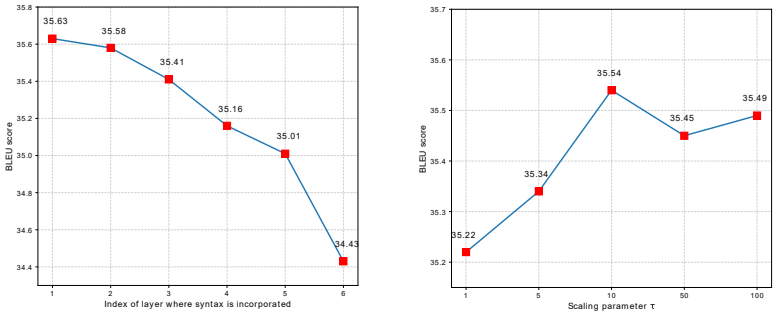
Table 3. The testing BLEU scores on the NC11-En/De, IWSLT14-De/En and WMT18-En-Tr dataset. All results are averaged over 5 random seeds and the improvements have statistical significance ($p<0.01$). Values with postfix * are from the original paper, models without * are our implementations.

Models	IWSLT14		NC11		WMT18
	De->En	En->De	De->En	En->De	En->Tr
LISA	34.97	29.06	27.51	25.65	-
LISA*	-	-	27.10	25.30	13.60
PASCAL	34.84	29.10	27.65	25.52	18.57
PASCAL*	-	-	27.40	25.90	14.00
SLA	34.59	28.71	27.29	25.03	17.94
Transformer	34.58	28.23	27.06	25.08	17.86
Transformer*	-	-	26.60	25.00	13.10
Our Model	**35.58**	**29.33**	**28.40**	**26.43**	**18.65**

5.3 Analysis

Location Sensitivity of Different Syntax Fig. 2(a) shows the variation of model performance under different incorporating layers of the constituency syntax. According to it, constituency grammar is sensitive to its incorporation location, with its guided performance rapidly dropping from 35.63 to 34.43 from shallow to deep layers. This indicates that constituency syntax information well aids model performance in the front layers while having a negative impact when existing in the latter layers. To explore the location sensitivity of different syntaxes, we conduct the same experiments on IWSLT-

De-En with PASCAL, which utilize dependency syntax. Table 4 displays the comparison results mentioned above, from which it turns out that dependency syntax is not as sensitive to incorporation locations, where the performance keeps around 34.6 whichever layer it is integrated. Therefore, the adaptive location matters more for constituency syntax than dependency syntax.



(a) (b)

Fig. 2. (a) The BLEU scores under the guidance of constituency syntax on different layers of Transformer. (b) The testing BLEU scores under different values of τ.

Table 4. The BLEU scores with different depths of dependency or constituency syntax-guided layers. Results are gained by averaging the last 10 epochs. The constituency syntax is incorporated in the manner of syntactic distance as introduced in section 4, while the dependency syntax is incorporated following PASCAL [2].

Syntax Layer	Constituency Syntactic Distance	Dependency Syntactic Distance
1	**35.63**	34.74
2	35.58	34.56
3	35.41	34.66
4	35.16	**34.75**
5	35.01	34.68
6	34.43	34.54

Effectiveness of Gating Network To validate the effectiveness of gating network, we incorporate syntactic structures into the first two heads of each layer without any gating on the IWSLT14-De-En dataset, and results are shown in Table 5. The model performance declines with the rise of the layer's index for single-layer incorporation, and cumulative incorporation of syntax information performs well in the first 2 layers but the performance declines from 35.65 to 35.52 as deeper layers are integrated with syntax information, which nicely aligns with our gating weights that distribute more heavily in the first two layers. Thus, our introduction of the gating network can well substitute the manual search of incorporation location with stronger interpretability.

Table 5. The BLEU scores with different constituency syntax-guided layers (our model). Results are gained by averaging the last 10 epochs.

Syntax Layer	BLEU	Syntax Layer	BLEU
1	**35.63**	1	35.63
2	35.58	1-2	**35.65**
3	35.41	1-3	35.60
4	35.16	1-4	35.63
5	35.01	1-5	35.51
6	34.43	1-6	35.52

Soft Parameter τ Syntactic distance is represented by a sequence of natural numbers that correspond with their rankings in computation, which results in 0-1 hard margins in the attention mask. To increase the robustness of the syntactic mask, the alternation can be smoothed in every row of attention masks by setting the scaling parameter τ larger than 1. The larger τ becomes, the smoother the attention mask will be. $\tau = 1$ corresponds to the original hard margin attention mask. Fig. 2(b) shows results of the BLEU score under different values of τ on the IWSLT14-De-En dataset, from which it can be concluded that the performance rises with a smoother attention mask while dropping again when it's too smooth. The model performs best when $\tau = 10$.

5.4 Ablation Study

Gating Network To verify the effectiveness of our gating network, we respectively try pure syntactic attention (abbreviated as 'Pure'), syntax guided attention averaged with raw attention (abbreviated as 'Avg'), and gating attention (abbreviated as 'Ours') on 5 datasets, and results are recorded in Table 6. Comparisons indicate that our gating network can consistently achieve the best BLEU score on all datasets compared with the other two settings. For some datasets like the IWSLT-En-De and NC11-De-En, without any gating weights, syntactic integrations by the simple average make performance respectively degrade 0.02 and 0.12 on the contrary.

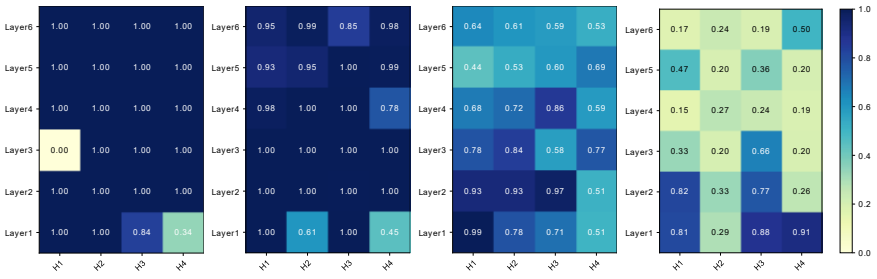
Table 6. The BLEU score under different manners of syntax incorporation on 5 datasets. Results are gained by averaging the last 10 epochs on the same random seed.

Datasets	Pure	Avg	Ours
IWSLT-De-En	35.48	35.65	**35.66**
IWSLT-En-De	29.44	29.42	**29.51**
NC11-De-En	27.89	27.77	**28.40**
NC11-En-De	25.02	26.20	**26.56**
WMT18-En-Tr	18.49	18.52	**18.75**

Regularization of Gating Network We drop our three regularization tricks: batch normalization, pre-warmup training, and syntax ignoring step by step, and the examples of visualization results on IWSLT-De-En dataset are shown in Fig. 3, with their corresponding performance comparisons in Table 7. When none of the regularization is added, nearly all gating values converge to 0 or 1 finally. As our training tricks are gradually introduced, the gating values tend to become smoother and stand out on two attention heads in the first 2 layers. It can be concluded that our regularization tricks on the gating structure can effectively conquer the self-reinforcing problem and help achieve better results.

Table 7. The BLEU scores under different combinations of regularization tricks on the IWSLT-DE-EN dataset.

Configuration	BLEU
All	**35.66**
BatchNorm + Pre-warmup	35.64
BatchNorm	35.50
Nothing	35.53



(a) (b) (c) (d)

Fig. 3. Ablation results for various regularization tricks. Each value in the figure represents a weight for incorporating SLR. 'H1' refers to the first attention head. "Layer1" refers to the first layer of Transformer. Without any regularization, the self-reinforcing problem would dedicate to completely SLR-filled attention weights. (a) Nothing. (b) Only batchNorm. (c) BatchNorm + pre-warmup. (d) All.

5.5 Model Size and Training Efficiency

The number of parameters of our model as well as its GPU training time consumption (in the unit of hours, abbreviated as 'GPU-hours') on each dataset is listed in Table 8. All trainings are conducted on NVIDIA GeForce RTX 3090 with CUDA version 12.0 and 8 CPU cores. The time consumption of our model on most datasets is around 2 hours.

Table 8. The model's capacity and its training time on all datasets.

Datasets	Parameter	GPU-Hours
IWSLT-De-En	36933680	1.8
IWSLT-En-De	36931632	1.8
NC11-De-En	40060976	2.5
NC11-En-De	40060976	2.5
WMT18-En-Tr	39991344	2.3

6 Conclusion

Building upon prior work that integrates constituency syntax into the Transformer by modifying attention matrices using SLR, this study introduces a novel architecture that adaptively learns the optimal amount of SLR guidance for each attention head across different layers, enhancing its flexibility and interpretability while reducing the burden of searching for the best incorporating locations. Experiments show that our model achieves a significant improvement compared with the vanilla Transformer and competitive results against other syntax-aware methods. Further studies disclose that constituency syntax is appropriate for integrations in shallow layers, which is distinctive from dependency syntax. Aligned with this finding, our adaptive gating weights yield an interpretable interface, where most syntactic guidance happens in the first two layers. There are also some limitations. Firstly, the effectiveness of syntax-aware self-attention relies on the quality of external parser, which is missing for most low-resource languages. Secondly, a few hyperparameters for regularization are still required, even though they can be accomplished in linear time complexity rather than the exponential complexity of searching for syntactic incorporating layers and attention heads. We would further dive into these problems.

References

1. Bengio, E., Bacon, P.L., Pineau, J., Precup, D.: Conditional computation in neural networks for faster models. arXiv preprint arXiv:1511.06297 (2015)
2. Bugliarello, E., Okazaki, N.: Enhancing machine translation with dependency-aware self-attention. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1618–1627. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.147, https://aclanthology.org/2020.acl-main.147
3. Clark, K., Khandelwal, U., Levy, O., Manning, C.D.: What does BERT look at? an analysis of BERT's attention. In: Proceedings of the 2019 ACL Work-shop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. pp. 276–286. Association for Computational Linguistics, Florence, Italy (Aug 2019). https://doi.org/10.18653/v1/W19-4828, https://aclanthology.org/W19-4828
4. Eigen, D., Ranzato, M., Sutskever, I.: Learning factored representations in a deep mixture of experts. arXiv preprint arXiv:1312.4314 (2013)

5. Goldberg, Y.: Assessing bert's syntactic abilities. arXiv preprint arXiv:1901.05287 (2019)
6. Hao, S., Zhou, Y., Liu, P., Xu, S.: Bi-syntax guided transformer network for aspect sentiment triplet extraction. Neurocomputing 594, 127880 (2024)
7. Harada, S., Watanabe, T.: Neural machine translation with synchronous latent phrase structure. Journal of Natural Language Processing 29(2), 587–610 (2022)
8. Hou, S., Kai, J., Xue, H., Zhu, B., Yuan, B., Huang, L., Wang, X., Lin, Z.: Syntax-guided localized self-attention by constituency syntactic distance. In: Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 2334–2341. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022). https://doi.org/10.18653/v1/2022.findings-emnlp.173, https://aclanthology.org/2022.findings-emnlp.173
9. Htut, P.M., Phang, J., Bordia, S., Bowman, S.R.: Do attention heads in bert track syntactic dependencies? arXiv preprint arXiv:1911.12246 (2019)
10. Huang, L., Sun, X., Li, S., Zhang, L., Wang, H.: Syntax-aware graph attention network for aspect-level sentiment classification. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 799–810. International Committee on Computational Linguistics, Barcelona, Spain (Online) (Dec 2020). https://doi.org/10.18653/v1/2020.coling-main.69, https://aclanthology.org/2020.coling-main.69
11. Jawahar, G., Sagot, B., Seddah, D.: What does BERT learn about the structure of language? In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3651–3657. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1356, https://aclanthology.org/P19-1356
12. Kai, J., Hou, S., Huang, Y., Lin, Z.: Leveraging grammar induction for language understanding and generation. arXiv preprint arXiv:2410.04878 (2024)
13. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. pp. 423–430. Association for Computational Linguistics, Sapporo, Japan (Jul 2003). https://doi.org/10.3115/1075096.1075150, https://aclanthology.org/P03-1054
14. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions. pp. 177–180. Association for Computational Linguistics (2007)
15. Li, Z., Zhou, Q., Li, C., Xu, K., Cao, Y.: Improving BERT with syntax-aware local attention. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 645–653. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.findings-acl.57, https://aclanthology.org/2021.findings-acl.57
16. Ma, C., Tamura, A., Utiyama, M., Sumita, E., Zhao, T.: Improving neural machine translation with neural syntactic distance. 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). pp. 2032–2037. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1205, https://aclanthology.org/N19-1205
17. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. pp. 55–60 (2014)
18. Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., Auli, M.: fairseq: A fast, extensible toolkit for sequence modeling. In: Proceedings of the 2019 Conference of

the North American Chapter of the Association for Computational Linguistics (Demonstrations). pp. 48–53. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-4009, https://aclanthology.org/N19-4009
19. Post, M.: A call for clarity in reporting BLEU scores. In: Proceedings of the Third Conference on Machine Translation: Research Papers. pp. 186–191. Association for Computational Linguistics, Belgium, Brussels (Oct 2018), https://www.aclweb.org/anthology/W18-6319
20. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/P16-1162, https://aclanthology.org/P16-1162
21. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538 (2017)
22. Shen, Y., Lin, Z., Jacob, A.P., Sordoni, A., Courville, A., Bengio, Y.: 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). pp. 1171–1180. Association for Computational Lin-guistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-1108, https://aclanthology.org/P18-1108
23. Shen, Y., Tay, Y., Zheng, C., Bahri, D., Metzler, D., Courville, A.: Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. arXiv preprint arXiv:2012.00857 (2020)
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15(1), 1929–1958 (2014)
25. Strubell, E., Verga, P., Andor, D., Weiss, D., McCallum, A.: Linguistically-informed self-attention for semantic role labeling. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 5027–5038. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018). https://doi.org/10.18653/v1/D18-1548, https://aclanthology.org/D18-1548
26. Tenney, I., Das, D., Pavlick, E.: BERT rediscovers the classical NLP pipeline. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4593–4601. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1452, https://aclanthology.org/P19-1452
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems 30 (2017)
28. Wang, Y., Lee, H.Y., Chen, Y.N.: Tree transformer: Integrating tree structures into self-attention. 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). pp. 1061–1070. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1098, https://aclanthology.org/D19-1098
29. Xie, R., Ahia, O., Tsvetkov, Y., Anastasopoulos, A.: Extracting lexical features from dialects via interpretable dialect classifiers. arXiv preprint arXiv:2402.17914 (2024)
30. Xu, Z., Guo, D., Tang, D., Su, Q., Shou, L., Gong, M., Zhong, W., Quan, X., Jiang, D., Duan, N.: Syntax-enhanced pre-trained model. 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). pp. 5412–5422. Association for

Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.420, https://aclanthology.org/2021.acl-long.420

31. Yang, J., Ma, S., Zhang, D., Li, Z., Zhou, M.: Improving neural machine translation with soft template prediction. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5979–5989. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.531, https://aclanthology.org/2020.acl-main.531
32. Zhang, M., Li, Z., Fu, G., Zhang, M.: Syntax-enhanced neural machine translation with syntax-aware word representations. 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). pp. 1151–1161. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1118, https://aclanthology.org/N19-1118
33. Zhang, Z., Wu, Y., Zhou, J., Duan, S., Zhao, H., Wang, R.: Sg-net: Syntax guided transformer for language representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(6), 3285–3299 (2020)