Syntax-guided Localized Self-attention by Constituency Syntactic Distance

Shengyuan Hou[1]* **Jushi Kai**[1]* **Haotian Xue**[1]*
Bingyu Zhu[2] **Bo Yuan**[2] **Longtao Huang**[2] **Xinbing Wang**[1] **Zhouhan Lin**[1]†
[1] Shanghai Jiao Tong University [2] Alibaba Group
{hsyhwjsr,json.kai,xavihart}@sjtu.edu.cn
{zhubingyu.zby,qiufu.yb,kaiyang.hlt}@alibaba-inc.com
lin.zhouhan@gmail.com

Abstract

Recent works have revealed that Transformers are implicitly learning the syntactic information in its lower layers from data, albeit is highly dependent on the quality and scale of the training data. However, learning syntactic information from data is not necessary if we can leverage an external syntactic parser, which provides better parsing quality with well-defined syntactic structures. This could potentially improve Transformer's performance and sample efficiency. In this work, we propose a syntax-guided localized self-attention for Transformer that allows directly incorporating grammar structures from an external constituency parser. It prohibits the attention mechanism to overweight the grammatically distant tokens over close ones. Experimental results show that our model could consistently improve translation performance on a variety of machine translation datasets, ranging from small to large dataset sizes, and with different source languages. [1]

1 Introduction

Although Transformer doesn't have any inductive bias on syntactic structures, some studies have shown that it tends to learn syntactic information from data in its lower layers (Tenney et al., 2019; Goldberg, 2019; Jawahar et al., 2019). Given the pervasiveness of syntactic parsers that provides high quality parsing results with well-defined syntactic structures, Transformers may not need to re-invent this wheel if grammar structures could be directly incorporated into it.

Prior to Transformer (Vaswani et al., 2017), earlier works have demonstrated that syntactic information could be helpful for various NLP tasks. For example, Levy and Goldberg (2014) introduced dependency structure into word embeddings, and Chen et al. (2017) uses Tree-LSTMs to process the grammar trees for machine translation.

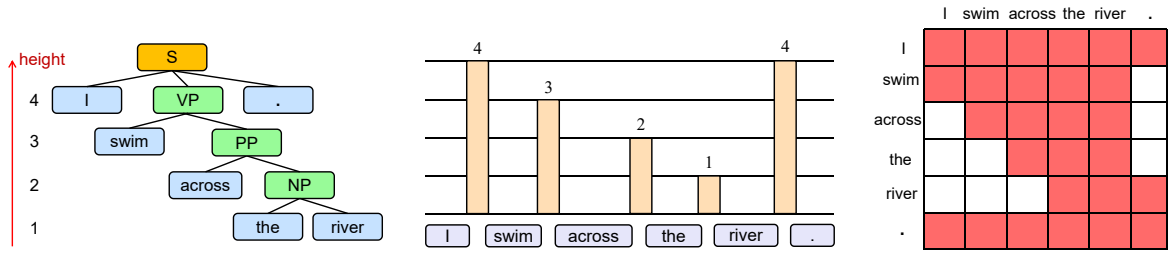
More recently, dependency grammar has been successfully integrated into Transformer in various forms. Strubell et al. (2018) improves semantic role labelling by restricting tokens to only attend to its dependency parent. Zhang et al. (2020) modifies BERT (Devlin et al., 2019) for named entity recognition as well as GLUE tasks (Wang et al., 2018) by adding an additional attention layer that allows every token to only attend to its ancestral tokens in the dependency parse tree. Bugliarello and Okazaki (2020) improves machine translation by constructing the attention weights from dependency tree, while Li et al. (2021) masks out distant nodes in the dependency tree from attention.

While the dependency grammar demonstrates the relation between nodes, the constituency grammar focuses more on how a sentence is formed in a merging way block by block. Constituency grammar contains more information about the global structure of a sentence in a hierarchical way, which we think will greatly improve global attention mechanism like self-attention in Transformers. Since constituency grammar doesn't directly reflect grammatical relations between words and introduces new constituent nodes, integrating it into Transformer becomes less obvious. Ma et al. (2019) explores different ways of utilizing constituency syntax information in the Transformer model, including positional embedding, output sequence, etc. Yang et al. (2020) uses dual encoders to encode both source text and template yielded by constituency grammar, at a cost of introducing a large amount of parameters.

In this work, we propose a syntax-guided localized self-attention that effectively incorporates constituency grammar into Transformers, without introducing additional parameters. We first serial-

* Equal contribution.
† Zhouhan Lin is the corresponding author.
[1]Our code is available at https://github.com/LUMIA-Group/distance_transformer

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2334–2341



(a) Constituency Grammar Tree (b) Syntactic Distance (c) Syntactic Local Range

Figure 1: (a) The constituency tree for the example sentence "I swim across the river.". (b) Its syntactic distances. (c) The attention mask reflecting the syntactic local ranges of each word For example, rather than attending to the whole sequence, "across" encourages attention towards *swim*, *the* and *river* while suppresses the others.

ize constituency trees through syntactic distance (Shen et al., 2018), and then select several attention heads as grammar-aware heads in which the attention ranges of each token are individually modulated according to their grammatical roles. the modulated attention ranges are named as *syntactic local ranges*, which prohibits the attention mechanism to overweight the grammatically distant tokens over close ones. Experimental results show that our model could consistently improve translation performance on a variety of machine translation datasets, ranging from small to large dataset sizes, and with different source languages.

2 Preliminary: Syntactic Distance

2.1 Definition

Syntactic distance (Shen et al., 2018) is a serialized vector representation of constituency grammar tree (Fig 1(a)) that is defined as:

Definition 2.1. ***(Syntactic Distance)*** *Given a length n sentence $S = (t_1, ..., t_n)$ and its constituency grammar tree T, in which the height of the lowest common ancestors of any pair of tokens t_i, t_j is noted as h^i_j. The syntactic distance $D = (d_1, ..., d_{n-1})$ of this sentence could be any vector of scalars with length $n-1$, which satisfies: $\forall i, j \in [1, n-1]$,*

$$sign(d_i - d_j) = sign(h^i_{i+1} - h^j_{j+1}) \quad (1)$$

Intuitively, syntactic distance D keeps the same ranking order as the sequence of $(h^1_2, h^2_3, ..., h^{n-1}_n)$, in which h^i_{i+1} is the height of the lowest common ancestors between pairs of consecutive words in the sentence (See Fig. 1(b)).

2.2 Generation of Syntactic Distance

The syntactic distance could be generated by recursively spliting the constituency tree in a top-down manner. According to the merging order of constituency syntactic tree, for any subtree T, the subtrees rooted by every child node of T must be constructed at first. Therefore, the merging of all of T's child nodes must take place afterwards. The syntax distance in all the subtrees of T can be calculated first, and then the maximum distance value plus 1 is the current merging distance order.

During preprocessing, the syntactic distance is calculated on different datasets respectively. For each sentence, we first concatenate all BPE word segmentations, then analyze the syntax tree structure using the Stanford corenlp toolkit (Manning et al., 2014), and calculate the syntactic distance according to the algorithm in Algorithm 1. When filling in the syntactic distance between words in BPE word segmentation, the lowest value is 0, and finally all syntactic distances are added by 1 to become a syntactic distance vector with a minimum value of 1. In the case of multiple sentences, we generate the syntactic distance of each sentence respectively, and then fill in the maximum 999 between different sentences, indicating that all sentences are merged at last.

3 Method

We are going to present a form of local self-attention that dynamically controls each word's attention range according to its syntactic role in the sentence (See Fig. 1(c)). Attention heads that incorporate this localized self-attention could significantly outweight the grammatically close tokens over distant ones, thus incorporate the syntax information as prior knowledge.

Algorithm 1 Syntactic Distance Generation

Input: Constituency Grammar Tree T
Output: Syntactic Distance $D = (d_1, ..., d_n)$

function DISTANCE(root)
  $\boldsymbol{d} \leftarrow [\,]$
  if root is not leaf **then** //Node can be split
    $d\_set \leftarrow \{\,\}$
    $maxd \leftarrow 0$
    for c in root's child **do**
      $\boldsymbol{c_d} \leftarrow Distance(c)$
      $maxc \leftarrow \max(c_{d,1}, ..., c_{d,|\boldsymbol{c_d}|})$
      $d\_set \leftarrow d\_set \cup \{\boldsymbol{c_d}\}$
      $maxd \leftarrow \max(maxd, maxc)$
    end for
    $nd \leftarrow maxd + 1$
    $\boldsymbol{d} \leftarrow [\boldsymbol{c_{d1}}; nd; \boldsymbol{c_{d2}}; ...; nd; \boldsymbol{c_{d|d\_set|}}]$
  end if
  return $\boldsymbol{d}$;
end function

3.1 Syntactic Local Range

It is widely believed that in a grammar tree, sibling words within a certain constituent is more correlated than words that are more distant (Klein and Manning, 2003). For example, in Fig. 1(a), for the word *the*, the word *river* locates to its right is more related than the word *I*. More generally, we define a *syntactic local range* for each of the words to mark a set of adjacent words as more syntactically correlated to it. For a given token t, we consider the words that are 1) direct siblings of t; or 2) left/right siblings of the constituent that has t on its left/right boundary respectively, as syntactically more correlated to t. Note that, depending on the structure of the grammar tree, the syntactic local range to the left of the word could be different with that to the right. Formally, we have the following definitions.

Definition 3.1. ***(Syntactic Local Range)*** *For a given token t, let f be its parent node in the constituency tree. Syntactic local range is defined as the concatenation of two ranges residing on t's left and right sides, corresponding to its pre-text and post-text directions, respectively.*

For the pre-text direction,

- *If t is not the leftmost child of f, then t's syntactic local range on its left side starts at f's leftmost child, and stops at t.*
- *If t is the leftmost child of f, back-trace its ancestors to find a nearest constituent v where v is not the leftmost child of its parent w. Then t's syntactic local range on its left side starts at w's leftmost child and stops at t.*

Similar rules apply for the pos-text direction except that the left/right directions are altered accordingly.

Algorithm 2 Inducing Syntactic Local Range

Input: Syntactic Distance $D = d_i (i \in [1, n-1])$
Output: Syntactic Local Range G

$G = \mathbf{0} \in \{0, 1\}^{n \times n}$
for i=1 to n **do**
  $b_l = \underset{j<i-1, d_j > d_{i-1}}{\arg\max} \{j+1\}$ // left boundary
  $b_r = \underset{j>i, d_j > d_i}{\arg\min} \{j\}$ // right boundary
  $G[i, b_l : b_r] = 1$
end for
return G

3.2 Inducing Syntactic Local Range from Syntactic distance

Intuitively, for a given token t, its syntactic local range keeps stretching on both pre-text and post-text directions until it hits a distance that is larger than t's on the corresponding side. We can represent all the syntactic local ranges for every words in a sentence as a masking matrix consisting of 0s and 1s (c.f. Fig. 1(c)), and compute it through Algorithm 2. It can be proved that the ranges generated by Algorithm 2 is identical to the syntactic local ranges defined by Definition 3.1:

Proof. Consider the token t_i in the sequence $(t_1, t_2, ..., t_n)$ with the syntactic distances $(d_1, d_2, ..., d_{n-1})$.

For the pre-text direction, let t_i's syntactic local range starts at t_l. According to Definition 3.1, t_l and t_i share a lowest common ancestor w. Let the height of w be h_w.

1. Since t_l is the leftmost child of w, we have $h_l^{l-1} >= h_w$.
2. Since t_i is either the immediate child of w, or the leftmost child of a constituent v who is the immediate child of w, we have $h_i^{i-1} = h_w - 1$.
3. For all tokens between t_l and t_i inclusively, w should also be their common ancestor. Thus we have $\forall j \in [l+1, i], h_j^{j-1} <= h_w - 1$.

Comprehensively, we have

$$\forall j \in [l+1, i-1], h_l^{l-1} > h_i^{i-1} >= h_j^{j-1} \quad (2)$$

According to Eq. 1, we could derive that

$$\forall j \in [l, i-2], d_{l-1} > d_{i-1} >= d_j \quad (3)$$

Thus validates line 3 in Algorithm 2.

Symmetrically, the proof for the post-text direction is obvious. □

Specifically, we first define the syntactic local pattern in a constituency tree in Definition 3.1, which is explicit and easy to understand. However, using syntactic distance can facilitate the computing of syntactic local pattern since it can be well paralleled. Therefore we turn to an alternative expression of Definition 3.1 using syntactic distance, and propose Algorithm 2 to generate mask from syntactic distance, which is also proved to be identical to the definition of syntactic local pattern.

3.3 Syntax-guided Self-Attention

Given the masking matrix G, we can incorporate it into self-attention through masked softmax to get a syntax-guided self-attention:

$$\text{Attn}(Q, K, V, G) = \text{S}_m\left(G, \frac{QK^T}{\sqrt{d_k}}\right)V \quad (4)$$

$$\text{S}_m(M, X) = \left[\text{S}\left(\frac{m_{ji}e^{x_{ji}}}{\sum_{k=1}^{n} m_{jk}e^{x_{jk}}}\right)\right]_{j=[1,\ldots,n]} \quad (5)$$

where $\text{S}_m(\cdot)$ and $\text{S}(\cdot)$ are masked softmax and softmax, respectively. And m_{ji} and x_{ji} are the element in the j-th row and i-th column of the M and X matrices. $[\cdot]_{j=[1,\ldots,n]}$ corresponds to vertically stack the elements yielded by the function inside it, whose input j being $1, \ldots, n$ respectively.

Moreover, since we want to encourage the attention within the syntactic local range and suppress those outside of it, rather than zeroing out the attention weights outside of the syntactic local range. To this end, we would still like the model to attend on tokens outside of the range. Thus, rather than using a crispy boundary, i.e., elements in G are either 0 or 1, we smooth it through the following steps.

First we define the *soft comparison* between d_i and d_j as

$$\alpha_i^j = \frac{\tanh((d_i - d_j)/\tau) + 1}{2} \quad (6)$$

datasets	Transformer	Ours
IWSLT14-De2En	34.56	**35.74**
IWSLT14-En2De	28.17	**29.28**
NC11-De2En	26.83	**27.67**
NC11-En2De	25.26	**26.19**
ASPEC-Ch2Ja	47.77	**48.34**
WMT14-En2De	27.3	**28.48**

Table 1: Test BLEU score on five datasets.

which yields 0 when $d_i \ll d_j$ and gradually increases to 1 as d_i increases, with τ controlling the softness.

We can simulate the attention range found by Algorithm 2 by multiplying α from the central token to both sides. i.e., entries in G can be approximated by

$$G_{ij} = \prod_{j \le t \le i-1} \alpha_t^{i-1}, j < i-1 \quad (7)$$

To initiate the continued product, mask values on the two secondary diagonals in G are set to 1.

We then use this syntactic-guided attention mask to augment some of the attention heads in the vanilla Transformer model, thus incorporates constituency grammar information into Transformer.

4 Experiments

4.1 Datasets and Experiment Settings

We implement our model based on Fairseq (Ott et al., 2019) toolkit The Stanford CoreNLP parser is utilized to generate constituency grammar tree, which is further used to calculate syntactic distance. All distance values are natural numbers with a minimum value of 1 and distances between sentences are set to 999 by default.

We evaluate our model on machine translation tasks of different languages. We use IWSLT-14 German-English on both directions (De2En and En2De), NC11 German-English on both directions, ASPEC from Chinese to Japanese (Ch2Ja), and WMT14 from English to German. Unless otherwise noted, we only modify the first encoder layer, with 3 heads for IWSLT14-En2De and NC11-En2De, 4 heads for NC11-De2En and IWSLT14-De2En, 2 heads for ASPEC-Ch2Ja and WMT14-En2De. Detailed settings and available in Appendix A.

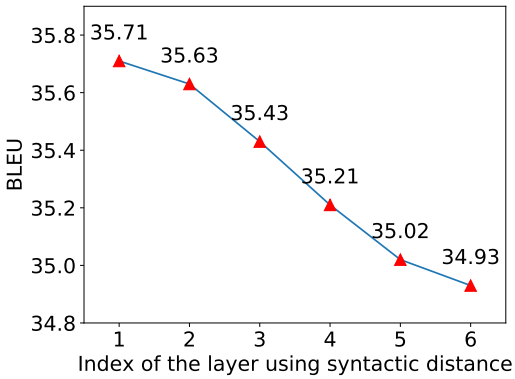


Figure 2: Layer-wise for IWSLT14-De2En

models	IWSLT14		NC11	
	De2En	En2De	De2En	En2De
baseline	34.56	28.17	26.83	25.26
1-head	35.65	29.20	27.58	25.87
2-head	35.68	29.21	**27.67**	25.99
3-head	35.71	**29.28**	27.56	**26.19**
4-head	**35.74**	29.23	**27.67**	26.08

Table 2: Head-wise for IWSLT14 and NC11.

4.2 Experimental Results

We compare our syntactic-guided model with Transformer (Vaswani et al., 2017) measuring by BLEU score[2]. All of the results are averaged over 5 independent runs on different seeds and the improvements have statistical significance ($p<0.01$). For the large WMT dataset, we average the test results over the last 5 checkpoints in a single run due to computation limits. Our model outperforms Transformer baseline model on all six machine translation datasets. Our model can achieve up to 1.18 improvements on these test beds. Detailed results are shown in table 1.

4.3 Ablation Study

We apply syntactic distance in different encoder layers of Transformer to get best utilization of syntactic distance. The results are shown in Fig. 2, where "x-layer" means that only 3 attention heads of x-th encoder layer use syntactic information. The model performs best with syntactic distance used in the first layer. Moreover, the deeper layer we add syntactic distance, the worse performance the model has. It can be concluded that top layers focus more on long range dependency of data.

[2]https://pypi.org/project/sacrebleu/

We also studied effects of different heads in using syntactic distance, as shown in table 2. "x-head" means we only use syntactic distance in x heads of the first encoder layer. We found that the utilization of different extent does not lead to great difference in performance.

5 Conclusion

In this work, we introduce constituency grammar into Transformer by using syntactic distance to guide the self-attention mechanism. We implement the complete pipeline of fusion by converting constituency tree to syntactic distance and finally generating syntactic local pattern. Experiments show that our method could consistently improve the performance of Transformer on a variety of machine translation tasks. We also excavate that the syntactic-guided attention achieve the best effectiveness in the bottom layer. Future work could expand to other NLU tasks and larger-scale pretrained models.

Limitations

Although the constituency-based syntactic local patterns implemented by syntactic distance appear to be advantageous for self-attention learning in seq2seq architecture, there are still some directions for further improvement.

Firstly, the constituency grammar information is generated explicitly using external parser, whereas most low-resource languages like Southeast Asian family lack sophisticated parsing toolkit. Specially consituency syntax tree bank is far rarer compared with dependency tree bank. Due to this, our experiments do not include low resource languages. To solve this problem, incorporating syntactic information implicitly without any annotation is a heated topic with great prospect. We will further explore this direction.

Also, most parsers are based on statistical methods so that generated grammar tree often entails noise and uncertainty. These hinder the wide applications of our method. A more modern parser like the Berkeley Neural parser with Bert or Roberta etc. embeddings would do better probably. It would be interesting to see whether the effect on the BLUE scores of using a parser has a higher accuracy that could presumably then provide the basis for an improved incorporation of syntactic representation.

Secondly, the utilization position of syntactic mask is a tough problem when it is manually tuned,

especially on large model configurations and large datasets. One alternative is to learn the usage of syntactic attention mask adaptively and softly, by which every attention head in every layer is a combination of original global attention and syntactic masked attention. The combination weight of every head is parameterized and learned throughout the training process.

Thirdly, the pretraining and finetuning architecture has been around in recent years for its generality and excellent performance compared with the task-specific model training from scratch. It's intuitive to explore whether our method could still achieve good performance on pretraining architecture such as BERT, even though some studies find that trained on a large corpus the pretraining model has inherently entailed syntactic structural information.

Fourthly, many studies focus on the usage of dependency syntax information to enhance the self-attention learning. The constituency grammar structure is naturally suitable for languages with strong local structure, while dependency grammar is suitable for languages with high flexibility of permutation. The difference between two kinds of syntax structure on different source languages are left for further study.

Acknowledgements

This work is sponsored by the National Natural Science Foundation of China (NSFC) grant (No. 62106143), and Shanghai Pujiang Program (No. 21PJ1405700). We would also like to thank the anonymous reviewers for their constructive comments and suggestions.

References

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, Online. Association for Computational Linguistics.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

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.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308. The Association for Computer Linguistics.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2021. Improving BERT with syntax-aware local attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 645–653, Online. Association for Computational Linguistics.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. 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)*, pages 2032–2037, Minneapolis, Minnesota. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. 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)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020. Improving neural machine translation with soft template prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5979–5989, Online. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. Sg-net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9636–9643. AAAI Press.

A Training Details

We perform experiments on six machine translation datasets and use the BLEU score to evaluate our model's performance. ISWLT14 EN/DE dataset is stemmed from TED talks. NC11 EN/DE is stemmed from news commentary. ASPEC CH/JA is stemmed from scientific paper excerpt corpus. WMT14 EN/DE is stemmed from European news. Sizes of different datasets are listed in Table3.

datasets	Train	Valid	Test
IWSLT14-De/En	160239	7283	6750
NC11-En/De	238843	2169	2999
ASPEC-Ch/Ja	672315	2090	2107
WMT14-En/De	4528223	3000	3003

Table 3: Sizes of Experimental Datasets

For Iwslt14 dataset preprocessing, we use Moses toolkit[3] for uncased word tokenization. Then we clean up the dataset, and only keep the pairs whose source-target length ratio is within 1.5 and the total length does not exceed 175. Next we use the subword NMT toolkit[4] for BPE subword tokenization with a shared dictionary of size 10k. Finally we randomly select 5% of training as the validation set, and merge dev2010, dev2012, tst2010, tst2011 and tst2012 into the test set. The size of the training, validation and test set are 160k/7.3k/6.7k respectively.

For NC11 dataset preprocessing, we still use Moses toolkit for uncased word tokenization. Then we clean up the dataset, and only keep the pairs whose source-target length ratio is within 1 and the total length does not exceed 80. Next we use the subword NMT toolkit for BPE subword tokenization with a shared dictionary of size 16k. Otherwise, we dropout the pairs which do not have a correct language. Validation and test set are newstest2015 and newstest2016 respectively. The size of the training, validation and test set are 239k/2.2k/3k respectively.

For ASPEC dataset preprocessing, we use Moses to extract statements. Then we use StanfordNLP-2014-01-01-segmenter[5] for Chinese and juman-7.0[6] for Japanese in word tokenization. Next we use the subword NMT toolkit for BPE subword tokenization with a Chinese dictionary of size 61k and a Japanese dictionary of size 46k. The size of the training, validation and test set are 672.3k/2k/2.1k respectively.

For WMT dataset preprocessing, subword-NMT and Moses are still ustilized for tokenization and

[3]https://github.com/moses-smt/mosesdecoder
[4]https://github.com/rsennrich/subword-nmt
[5]https://nlp.stanford.edu/software/segmenter.shtml
[6]https://nlp.ist.i.kyoto-u.ac.jp

dictionary built-up. The size of BPE shared dictionary is 40000. Non-printing characters are removed and all punctuations are normalized. The source-target length ratio and the total length are limited within 1 and 250 respectively. We sample 1% of training data as validation set and test sets are directly downloaded.The size of training, validation and test set are 4528k/3k/3k respectively.

Manual Tuning is performed for hyper-parameters tuning on the basis of test BLEU score for the best checkpoint. We use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-8}$. We use label smoothing of value $\epsilon_{ls} = 0.1$ and weight decay of 0.0001. By default, the attention-dropout is set to 0.2, dropout is set to 0.3, except that we set the attention-dropout to 0.1 in ASPEC and set the attention-dropout and dropout both to 0.1 in WMT. We take the inverse square root learning rate scheduler and set the peak learning rate as 1e-3 with 4000 steps linear warm-up and 1e-7 initial learning rate. The scaling parameter τ is set to 10. The model has a hidden dimension of 512, 6 encoder layers and 6 decoder layers with 4 attention heads per layer, except for WMT dataset with 8 attention heads per layer. Hidden layer size for FFN is set to 2048 in ASPEC and 1024 in other datasets. For IWSLT14 and ASPEC, each update has up to $8192 \times 2 \times 1$ tokens(max_tokens$\times$gpu_num$\times$num_updates). For NC11, each update has up to $8192 \times 2 \times 2$ tokens. We use 8 80G-A100 GPUs from distributed clusters to train our model on WMT14 dataset with CUDA version 11.1 and 2 40G-A100 GPUs for other datasets with CUDA version 11.2.

datasets	Ours	Transformer
IWSLT14-De2En	80min	70min
IWSLT14-En2De	80min	80min
NC11-En2De	80min	80min
NC11-De2En	80min	80min
ASPEC-Ch2Ja	12h	12h
WMT14-En2De	9h	8.5h

Table 4: Training time of Different Datasets

Since our incorporation of syntactic information is based on explicit calculation of syntactic distance and local attention pattern mask by external syntactic parser, we do not introduce extra parameters compared with original Transformer model. The number of parameters is 50M for configuration of WMT dataset, 99M for ASPEC dataset and 30M for the other datasets.

The training time of Transformer and our distance enhanced model are listed as in table 4. Basically, our model costs about the same time for training as Transformer.