Transformer with Syntactic Position Encoding for Machine Translation

Yikuan Xie[1], **Wenyong Wang**[1], **Mingqian Du**[2], and **Qing He**[3]

[1]University of Electronic Science and Technology of China
[2]The Second Research Institute of CAAC
[3]The Third People's Hospital of Chengdu City
xieyikuan@std.uestc.edu.cn, wangwy@uestc.edu.cn
dumingqian@caacetc.com, 123hq@163.com

Abstract

It has been widely recognized that syntax information can help end-to-end neural machine translation (NMT) systems to achieve better translation. In order to integrate dependency information into Transformer based NMT, existing approaches either exploit words' local head-dependent relations, ignoring their non-local neighbors carrying important context; or approximate two words' syntactic relation by their relative distance on the dependency tree, sacrificing exactness. To address these issues, we propose global positional encoding for dependency tree, a new scheme that facilitates syntactic relation modeling between any two words with keeping exactness and without immediate neighbor constraint. Experiment results on NC11 German→English, English→German and WMT English→German datasets show that our approach is more effective than the above two strategies. In addition, our experiments quantitatively show that compared with higher layers, lower layers of the model are more proper places to incorporate syntax information in terms of each layer's preference to the syntactic pattern and the final performance.

1 Introduction

Over the past few years, end-to-end neural machine translation (NMT) has shown remarkable progress, achieving promising results on various machine translation tasks. Although NMT models can perform well even only trained on parallel corpus, they have been found to still benefit from external linguistic knowledge (e.g., lexical analysis and parsing) (Sennrich and Haddow, 2016), and the same finding was also observed in other tasks (Strubell et al., 2018; Zhang et al., 2019). Therefore, several studies have started to to investigate how to incorporate dependency information into NMT based on Transformer, which is currently the most advanced end-to-end machine translation backbone.

There exist two broad categories of approaches to integrating a sentence's dependency structure into the Transformer. The first category of approaches (Bugliarello and Okazaki, 2020; Strubell et al., 2018; Bastings et al., 2017) takes the dependency tree as a general graph structure data. A graph neural network or a sublayer with a similar mechanism was introduced at the bottom of the Transformer's encoder to represent and encode the dependency relations among the words in the sentence. In this paradigm, however, a word focuses only on its immediate neighbors and ignores nodes that are multi-hop away from it, which may carry important context that helps disambiguate and enrich the current word's representation in terms of syntactic structure.

In contrast, methods from the second category seek to model two words' syntactic relation relying on their distance from the dependency tree. The distance can be two words' *relative depth* on the tree or the *length of the path* between the two nodes (Wang et al., 2019a; Omote et al., 2019). This strategy can be seen as extended relative position representations (Shaw et al., 2018) where the relative distance is defined based on the tree. The advantage of this strategy is that the syntactic relations between a word and all other words can be modeled, not just those of its immediate neighbors. Nevertheless, this approach is also unsatisfactory because the distance on the dependency tree is a simple proxy for the syntactic relationship between two words and does not represent how a word reaches another word on the dependency tree. Therefore, this type of approach has limitations in expressiveness.

For these considerations, we propose to model the position of the word on the dependency tree in addition to its normal sequential position in the input text. We use the path from the root node to a

Proceedings of Recent Advances in Natural Language Processing, pages 1536–1544
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_172

word on the dependency tree to represent the global syntactic position of the word. This representation facilitates the modeling of syntactic relations between two words at arbitrary distances while enabling the model to learn more powerful correlation functions by exploiting dependency labels rather than simply relying on the distance of word pairs.

Our main contributions are as follows:

- We propose a new scheme, global positional encoding for syntax (GPS), to integrate the dependency tree into the Transformer's encoder, enabling efficient word pairs' syntax correlation modeling (Section 3).
- Experimental results demonstrate that our method outperforms both the Transformer baseline and other two competitive models in terms of BLEU score (Section 4.2).
- We found that when external syntax annotation is combined at lower layers of the encoder, it gives better results and has larger weights; and the weights, which indicate a layer's reliance on syntactic information, declines rapidly near the output layer of the encoder (Section 4.3).

2 Related Work

Previous studies showed that end-to-end NMT models could be further improved by combining source-side or target-side syntax structure into RNN encoder-decoder framework (Aharoni and Goldberg, 2017; Wu et al., 2017; Eriguchi et al., 2016). Since the Transformer has become the de facto standard architecture for NMT, researchers have also begun to explore strategies to incorporate syntax information into Transformer-based NMT.

For phrase structures, Currey and Heafield (2019) used source-side linearized constituency parses to improve Transformer-based NMT by multi-task learning. Ma et al. (2019) used neural syntactic distance (Shen et al., 2018) for constituent parsing as input and output sequence.

For dependency grammar, researchers mainly focused on how to integrate the dependency structure into the Transformer. As mentioned in the introduction, there are two different viewpoints to look upon this problem. The first line of works extended the relative positional encoding (Shaw et al., 2018) defined on sequence to tree structures. Omote et al. (2019) defined the relative distance between two words on the dependency tree in terms of their relative depth. Wang et al. (2019b) used the length of the shortest path between two nodes on the dependency tree as their relative distance, and the depth of each node as their absolute position, in addition to each word's normal sequential position. Another perspective is to take the dependency tree as a general graph structure, focusing on local connection relations and apply methods from graph neural networks (or use similar mechanisms). Bastings et al. (2017) use a graph-convolutional networks to encode the source syntax structure, and more recently, Bugliarello and Okazaki (2020) directly encode the dependency structure inside the Transformer's self-attention module, achieving state-of-the-art results. By contrast, we neither treat the dependency tree as a simple sequential structure extension nor as general graph data. We seek to represent each word's role on the dependency tree, where the representation is tailored for the dependency grammar's tree structure, making it sensible to the syntactic connection with other words.

Our work is also inspired by researches on making Transformer sensible to structured input. Shiv and Quirk (2019) proposed a tree-structure positional encoding to make the Transformer sensible to source code's tree structure in the code processing task. The main idea is to define the path from the root node to a leaf node as the leaf node's position on the tree. We extend this idea from unlabeled directed trees to labeled trees and break the limitation of the tree depth and the number of child nodes. Moreover, we choose to use a sequence encoder instead of concatenating sub-vectors to compute the final path embedding, which utilizes the whole vector space and enables more powerful modeling of the interaction between two path embeddings.

3 Model

In this section, we first recap the basic Transformer architecture and then describe the proposed global syntactic position encoding mechanism.

3.1 Transformer Architecture

Transformer (Vaswani et al., 2017) is an encoder-decoder architecture composed of stacked encoder and decoder layers. Each encoder layer has three main components: residual connections, feed-forward layer, and self-attention. The decoder layer has extra decoder-encoder attention to access the output of the encoder. Among these components,

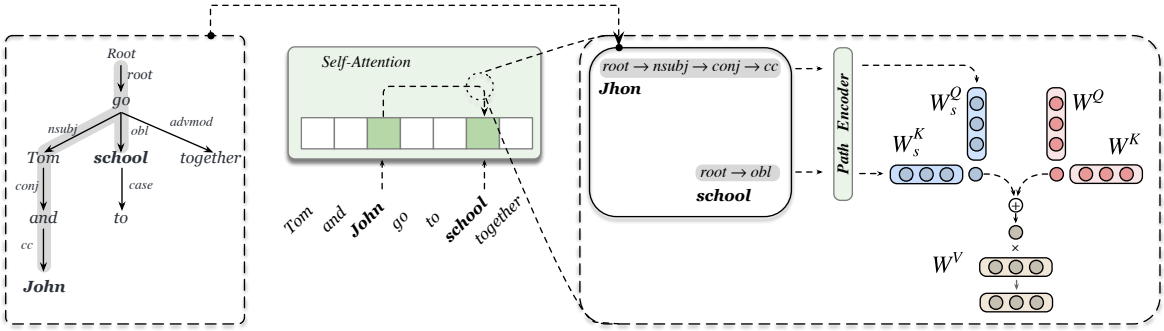

Figure 1: Illustration of our model's attention of the word 'John' to the 'school'. The dotted box on the left denotes the dependency analysis of the sentence "Tom and John go to school together," and the series of arrows with a gray background represents the path. We extract the two words' paths on the dependency tree and embed the paths to form query embedding and key embedding, which are then used to calculate the attention score.

the self-attention mechanism is the key feature that sets Transformer apart from CNN-based and RNN-based sequence to sequence (seq2seq) models. Self-attention is used for improving a word's representation by focusing on critical context words that help to enrich and disambiguate it.

Given an input sequence $x = \{x_0, ..., x_n\}$, self-attention outputs a new sequence $z = \{z_0, ..., z_n\}$. Each word's representation in the new sequence are contextualize by a weighted sum over all words in the old sequence.

$$z_i = \sum_j^n \alpha_{ij}(x_j W^V) \tag{1}$$

Each weight, α_{ij}, is normalized by a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})} \tag{2}$$

e_{ij} is computed by the dot product of $word_i$'s query vector and $word_j$'s key vector

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}} \tag{3}$$

where $\sqrt{d_z}$, a scale factor, reduces the peak of the attention distribution to alleviate the vanishing gradients problem.

3.1.1 Sequential Positional Encoding

Transformer's self-attention can access information directly from other words no matter how far away they are from the current word, making it easier to model long-range dependencies. However, unlike RNN's intrinsic order-variant forward process, the self-attention module is computationally insensible to the word order. Thus we need to put the word order information into the representation of the input sequence. In practice, we often simply add the positional embedding and input word embedding together as inputs to the model. Transformer's default positional embeddings has the following formulation:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right)$$

where pos is the index of a word in the sequence and i is the dimension of the positional encoding vector.

3.2 Positional Encoding for Dependency Structure

A sentence exists as a sequence of words, but it also has structural organization. Dependency parsing describes a sentence as a labeled directed tree, whose vertices correspond to words, and labeled edge denotes head-dependent relation between words. Besides each word's sequential position, we would like to represent their positions on the dependency tree to exploit prior linguistic knowledge behind the sentence.

We follow similar steps of encoding position in sequential structure to represent a word's position on the dependency tree. We need first to index a word and then transform the index into a vector, also called positional embedding, and the computation between two positional embeddings should reflect their relationship on the target structure. For sequential structure, the relation is the distance, and for the dependency tree, the relation is the grammatical relation denoted by how a node reaches another node. Therefore, we can represent

a words' tree position through the path from the root node to it. Figure 1 showing an example, *John* can be represented by a sequence of grammatical labels $[root \rightarrow conj \rightarrow cc]$ and *school* can be represented by $[root \rightarrow obl]$.

In this setting, the syntactic relation between two words can be inferred from their position representation on the syntactic tree.

To acquire the final syntactic positional embedding, we only need to vectorize the path. First, we replace all the labels with index from the dependency label vocabulary, then we encode the sequence into a vector, as if we are dealing with a sequence of words. Any neural network that can transform a sequence into a vector fits this job well, and we take LSTM as our syntactic position encoder:

$$s_i = LSTM(\boldsymbol{p_i})$$

where s_i is the syntactic positional embedding for x_i, and $\boldsymbol{p_i} = \{label_0, ..., label_n\}$ is the path from root to x_i on dependency tree.

3.3 Combining the Two Positional Embeddings

Once we can represent a word's position on the dependency tree, the next question is how to enrich a word's hidden representation by its position on the syntactic tree, in addition to its position in the sequence. Following Transformer's initial setup to incorporate sequential positional embedding, we may add the syntactic positional embedding into the input word embedding. However, this approach suffers from two drawbacks. First, when summing up heterogeneous embeddings, we are implicitly model correlation between different types of embedding in the self-attention module, which is noticed by Ke et al. (2021). To realize the problem here, we can split the hidden representation of a word x_i into the word embedding w_i, positional embedding p_i, and syntactic positional embedding s_i, and see how the self-attention computes the attention weight:

$$\begin{aligned} e_{ij} = &(\underbrace{w_i + s_i + p_i}_{x_i})W^Q) \cdot \\ &(\underbrace{w_j + s_j + p_j}_{x_j})W^K)^T / \sqrt{d_z} \end{aligned} \tag{4}$$

After expanding the above equation, we can see that the attention weight of word i to word j, denoted by e_{ij}, consists of the correlation between heterogeneous embeddings, e.g., the correlation between syntactic positional embedding and sequential positional embedding, which is not reasonable.

More importantly, when a source sequence contains multiple sentences, mixing the syntactic positional embeddings with word embedding allows two words from different sentences to interact with each other syntactically in the self-attention module. However, two words' syntax annotations are supposed to affect each other only within a sentence.

Therefore, we decouple the syntactic positional embedding and word embedding:

$$\hat{e_{ij}} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}} + \frac{\left(s_i W_s^Q\right)\left(s_j W_s^K\right)^T}{\sqrt{d_z}} \tag{5}$$

$$x_i = w_i + p_i \tag{6}$$

where x_i is the word representation as in Transformer's typical setting, and s_i denotes the syntactic positional embedding of word i. In this manner, we can eliminate the unnecessary correlation between heterogeneous embeddings and avoid undesired syntactic correlation for two words from different sentences by simply mask the second term in equation (5).

4 Experiment

4.1 Experimental Setup

Model and Baselines. We compare our approach with a strong baseline, Transformer, and other two types of dependency information augmented strategies: (1) Parent scaled self-attention (PASCAL) proposed by Bugliarello and Okazaki (2020), in which a word's attention distribution is scaled to have larger weight on its head word. (2) Absolute and relative structural position proposed by Wang et al. (2019b), which augment encoder by each word's depth and word-pair's relative distance on the dependency tree. We implement our models and reimplement Transformer with structural position (Wang et al., 2019b) based on the Fairseq (Ott et al., 2019) toolkit. We deploy the syntactic positional encoding at the bottom layer for our model and report the performance in the main results section. See section 4.3 for further analysis of this design choice.

Data. We use datasets with different sizes and source languages to evaluate the efficiency of our

Dataset	Train	valid	Test
NC11 En-De	238,843	2,169	2,999
NC11 De-En	238,843	2,169	2,999
WMT16 En-De	4,281,379 (filtered)	2,169	2,999
WMT16 En-De	5,852,458	2,999	3,004

Table 1: Data statistics

approach. For English → German translation task, we train on WMT16 and WMT17 corpus and use newstest2015/ newstest2016 as validation/ test sets for WMT16, newstest2016/newstest2017 for WMT17's validation/test sets. Following Bugliarello and Okazaki (2020), we remove samples with incorrect source language in WMT16 dataset. We also train on News Commentary v11 (NC11) for German → English and English → German tasks. See Table 1 for the dataset statistics for our experiments.

For data processing, we follow the same steps of Vaswani et al. (2017). We tokenize and split data with Moses (Koehn et al., 2007), using byte pair encoded (BPE) with shared vocabulary for German and English. We use Stanford CoreNLP parser (Manning et al., 2014) to acquire dependency trees of source sentences.

Training. For all models, we use Adam optimizer and learning rate scheduler with 8,000 warm-up steps. We train the model 100K steps on the WMT16 and WMT17 datasets and 20K steps on the NC11 dataset, which is much smaller than the previous two large-scale datasets. For Transformer and our path-encoder, we use $dropout = 0.1$ on WMT16 and WMT17, and $dropout = 0.3$ on NC11 dataset. We use label smoothing $\epsilon = 0.1$ for all experiments. During inference, we use beam search with $beam\_size = 4$ and length penalty $\alpha = 0.6$. We report the final results using SACREBLEU (Post, 2018).

4.2 Main Results

We can see from Table 2 that in all three syntax-augmented NMT systems, syntax information improves the performance of the baseline model at least 0.5 BLEU points.

On the other hand, although the distance-based approach (Structure Position) can theoretically model the syntactic relationship between any pair of words, it performs slightly inferior (0.1−0.3 BLEU points) than the PASCAL model, which only represents local head-dependent relations; this may be caused by information loss when using distance to approximate the syntactic relation between two words. In contrast, our strategy represents each word's global position on the syntax tree, facilitating efficient syntax relation modeling between non-local word pairs on the dependency tree. This further leads to consistent improvements over other models.

4.2.1 Effect of Sentence Length.

The effect of sentence length on translation quality has been an important research question, motivating us towards better model architecture and helpful training tricks, e.g., reversed input order, bidirectional-encoder, and attention. We make a further comparison between our model and the baseline concerning the sentence length. We collect five groups of test examples according to their source sentence length {0-20, 20-30, 30-40, 40-50, 50+}, and evaluate each group's BLEU score, which is plotted in Figure 2. We can observe a general trend in Figure 2 that the syntactic information helps the model achieve better translation especially on longer sentences, which are likely to have more complex structures and rich connections among the words in them.

4.3 Effects of Syntax at Different Layers

Previous researches empirically found that combining the syntax information at the bottom layer of the encoder gives the best result (Strubell et al., 2018; Bugliarello and Okazaki, 2020). In this subsection, we measure how our model's performance changes over the layer that we choose to combine the syntactic position, and quantify the syntax's impact on each layer's self-attention module.

First, we change the layer used to combine the syntactic position and train the model on the NC11 De-En dataset. We present the performance over the selected layer in Table 3. Consistent with previous studies, we observed that combining syntax structure at lower layers gives better results and achieves the best at the bottom layer. However, though the syntactic information brings performance gains no matter at which layer we combine it, these improvements are not complementary. When we combine the syntactic position at all six layers of the encoder, we observe comparable results than only combining it at the first layer of the encoder while accompanied by severe overfitting and much more training time.

Model	NC11 En-De	NC11 De-EN	WMT16 En-De	WMT17 En-De
Transformer	25.0	26.6	33.0	25.5
PASCAL	25.9	27.4	33.9	26.1
Structure Position	25.7	27.1	33.8	26.0
Ours	**26.0**$^{\Uparrow}$	**27.6**$^{\Uparrow}$	**34.1**$^{\Uparrow}$	**26.3**$^{\Uparrow}$

Table 2: Performances on the test set of three datasets in terms of BLEU score, with $^{\Uparrow}$ denotes statistical significance over the *Transformer* ($p < 0.05$) through boot strap resampling (Koehn, 2004)

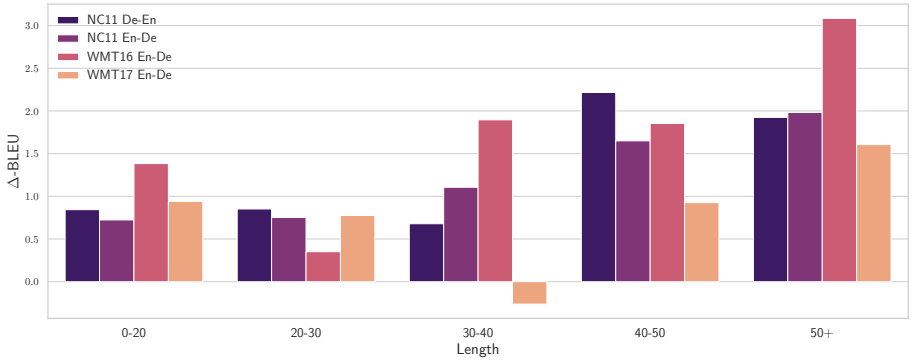

Figure 2: BLEU improvements on four datasets over the length of the source sentence.

In our setting, the syntactic information exerts influence directly on the self-attention distribution. Nevertheless, it remains unclear to what extent the model relies on the supplied syntax information, whose answer would provide us more interpretability and understanding of what is happening inside the model. To figure out this problem, we define the Average Weight of Syntax Logits (AWSL) as a metric to indicates the weight of syntax in the self-attention module. For each sample and attention head, we compute the weight of syntax logits as:

$$w = \frac{1}{N^2} \sum_i \sum_j \frac{|e_{ij}|}{|e_{ij}| + |e^s_{ij}|} \tag{7}$$

where N is the number of words in the sentence, e_{ij} and e^s_{ij} denotes the normal attention weight and the syntax correlation weight of word i to word j, respectively. We then average the weight w for each sample and attention head to compute $AWSL_l$ for layer l, as plotted in Figure 3.

Through the lens of AWSL, we can observe that the model relies on the abstract syntax patterns more at the lower layers. Such reliance becomes gradually smaller as the layer becomes deeper but drops sharply near the output layer.

One possible explanation for the two trends about model performance and feature importance is that combining syntax information at lower layers not only leads to better utilization across all

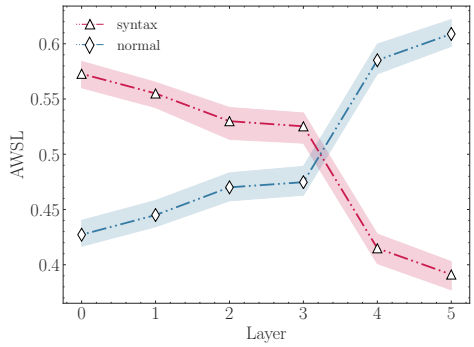

Figure 3: The red line shows how the Average Weight of Syntax Logits (AWSL) varies over the layer we choose to combine syntax structure, and the blue line denotes the weight of the model's standard attention module.

subsequent layers but also reaches a closer match in terms of the abstraction level between the layer and the supplied information. We note that this finding is in line with previous studies on probing the hidden representations of the deep language model (Raganato and Tiedemann, 2018; Tenney et al., 2019; Vig and Belinkov, 2019), which found that the model tends to encode more syntactic features in lower layers while more complex semantic features in higher layers.

Layer None	0	1	2	3	4	5	1-6
BLEU 23.8	24.7	24.5	24.5	24.0	24.1	24.0	24.7

Table 3: Model performance over the layer chose to combine syntactic position.

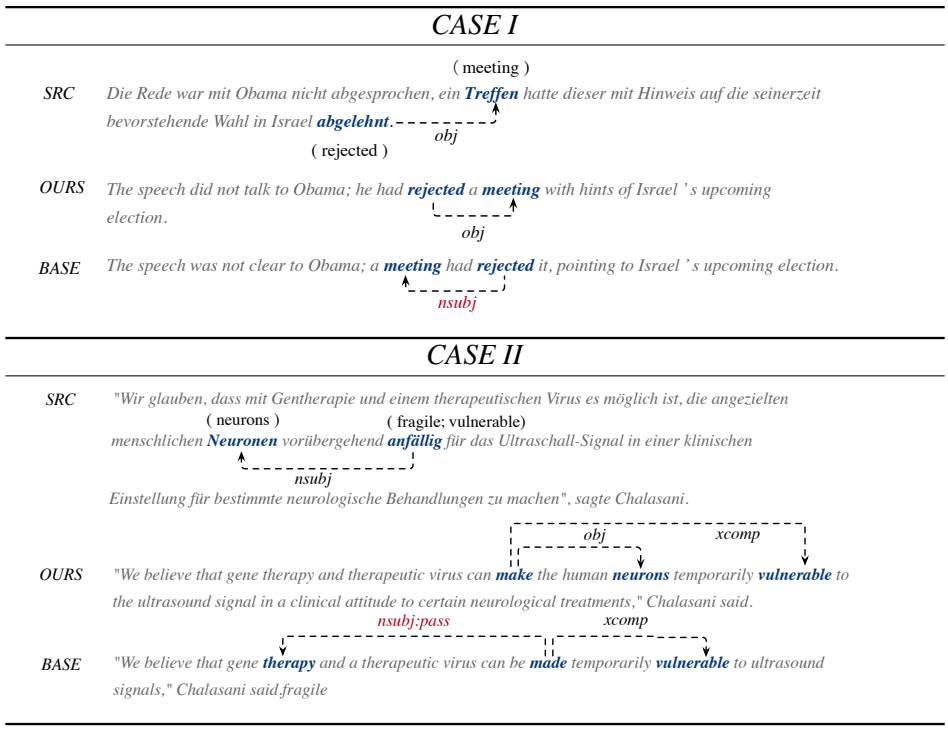

Table 4: We can see through the two cases that the baseline model generates syntactically incorrect translation due to the misunderstanding of the source sentence's syntactic structure, and the supplied syntax structure helps it achieve more faithful translation.

4.4 Case Study

Table 4 presents several representative cases that indicate how syntactic knowledge helps the model achieve better translation, while linguistically-uninformed baseline makes mistakes. In the first case, the *meeting* is the object to be *rejected*. However, the baseline takes *meeting* as nsubj (nominal subject) of the verb *rejected*, while our model correctly identified the *meeting* as the verb's object. A more complex example is the second case, where the adjective *anfällig* (means fragile) governs the nominal *Neuronen* (means neurons). The baseline wrongly and indirectly connects *therapy* and *vulnerable*, establishing misleading connection "*therapy* is made *vulnerable*", while our model correctly set up the second-order relation between *neurons* and *vulnerable*: "make *neurons vulnerable*".

5 Conclusion and Future Work

In this paper, we propose a global position encoding scheme for the dependency tree. We leverage the dependency labels and represent each word's syntactic role on the dependency tree using the path, which enables efficient syntactic correlation modeling between any pair of words in a single layer. In the experiment, we show that our model outperforms other syntax augmentation strategies. In addition, we quantitatively analyze the model's reliance on syntax information and show that the model pay more weight and achieves better performance when we combine syntax information at lower layers.

For future works, the following problems are worth noting:

1. Since it is commonly known that off-the-shelf parser's performance may drop dramatically when facing out-of-domain data, it is important to assess the parser's accuracy on different machine translation benchmarks and evaluate those syntax augmented NMT systems' tolerance of parsing errors.

2. Previous works have shown that human-

designed patterns are not the only option to establish dependency relations among words; trees induced from the Pre-trained Language Models also exhibit promising results in downstream applications (Wu et al., 2020; Dai et al., 2021). The induced tree is an attractive solution, especially when considering the expensive annotation and domain adaption problem for supervised dependency parser. Therefore, it is desirable to compare the performance of model using trees produced by the supervised parser and trees induced from the Pre-trained Language Models.

6 Acknowledgement

The work was supported by key project of NSFC-Civil Aviation Joint Fund(U2033212). The authors thank the three anonymous reviewers for their valuable comments.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *ACL*, pages 132–140, Vancouver, Canada. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

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.

Anna Currey and Kenneth Heafield. 2019. Incorporating source syntax into transformer-based neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 24–33, Florence, Italy. Association for Computational Linguistics.

Junqi Dai, Hang Yan, Tianxiang Sun, Pengfei Liu, and Xipeng Qiu. 2021. Does syntax matter? A strong baseline for aspect-based sentiment analysis with roberta. *CoRR*, abs/2104.04986.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. 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*, pages 177–180, Prague, Czech Republic. 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.

Yutaro Omote, Akihiro Tamura, and Takashi Ninomiya. 2019. Dependency-based relative positional encoding for transformer NMT. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, Varna, Bulgaria. INCOMA Ltd.

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 NAACL-HLT 2019: Demonstrations*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the*

2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. 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.

Vighnesh Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Hong Wang, Xin Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019a. Self-supervised learning for contextualized extractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2221–2227, Florence, Italy. Association for Computational Linguistics.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019b. Self-attention with structural position representations. 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 1403–1409, Hong Kong, China. Association for Computational Linguistics.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, Vancouver, Canada. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. 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 4568–4578, Hong Kong, China. Association for Computational Linguistics.