Augmenting Transformers with Enhanced Dependency Structures by Treating Relations as New Words

Jyun-Wei Chen[1], Shiou-Chi Li[1,2], and Jen-Wei Huang[1] ✉

[1] Department of Electrical Engineering,
National Cheng Kung University, Tainan, Taiwan
a0929551932@gmail.com, jwhuang@mail.ncku.edu.tw
[2] Institute of Computer and Communication Engineering,
National Cheng Kung University, Tainan, Taiwan
sp822543@gmail.com

Abstract. One of the Transformer's weaknesses is its ignorance of the structural information in the input sentence. Several methods have been proposed to enhance the Transformer's structural awareness by integrating grammatical structures, such as the dependency tree, into the self-attention module. However, previous methods often have a one-way interaction problem in which the learned information only flows from the relation embedding to the word embeddings during a forward pass. We propose to solve this one-way interaction problem by treating the relation labels in the dependency tree like the words in the sentences. We present the edge relation labels as independent tokens in the input sequence. Then, we could utilize the attention masking mechanism to inform the Transformer model about the structural information by only allowing attention calculations between some of the words and relations. Experiments showed that our approach could improve language models' performances in various downstream fine-tuning tasks.

Keywords: Dependency Tree Transformation · Dependency Tree · Transformer

1 Introduction

After the Transformer[11] architecture was introduced, its high parallelizability and ability to capture long-range dependency in a sequence allowed it to achieve many state-of-the-art results over traditional deep learning methods such as CNN and LSTM. As the large-scale language model pre-training techniques are widely adopted, the Transformer has become the most used architecture in the NLP research area. However, the Transformer has the same disadvantage as traditional sequential models, such as LSTM, in modeling sentence structures. Despite the fact that Transformer's self-attention mechanism excels in dynamically deciding the importance of each word, it pays less attention to the structure of the sentence.

In order to improve the Transformer's structural awareness while maintaining its efficiency, several methods have been proposed to extend the self-attention mechanism to allow the transformer model to take advantage of the structural information embedded in the sentence. Graph2Graph Transformer (G2GTr)[7] added new key and value vectors to represent each edge relation and added the attention results together while computing the attention weights and the value vector outputs. Dependency Transformer (DT)[5] proposed directly manipulating the attention weight scores for the word pairs connected in the dependency tree.

However, both G2GTr and DT only considered adding relation information in a one-way fashion. In other words, the models inject learned information stored in relation embeddings into the attention calculation but not in the other direction. The relation embeddings are only updated during backpropagation in the training stage. We call the above problem the one-way interaction problem. We use data flow diagrams to illustrate the one-way interaction problem of G2GTr and DT in Fig. 1(a) and 1(b).

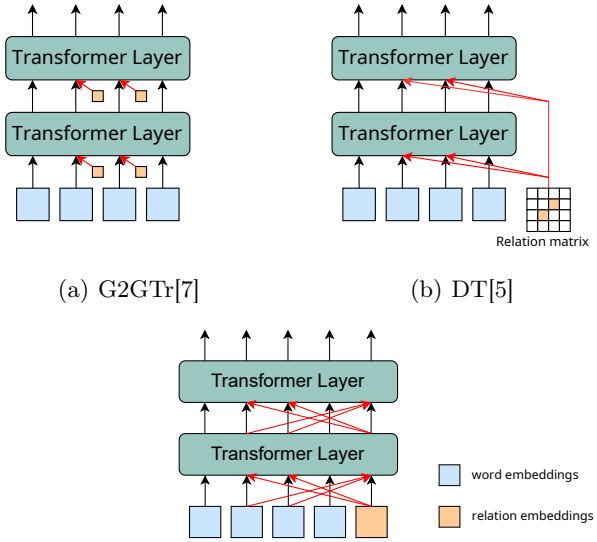
In order to solve the one-way interaction problem, we propose to look at the structure of the dependency tree differently. We proposed the Dependency-aware Attention Masking (DAM) framework. The data flow of the DAM framework is shown in Fig. 1(c). In DAM, we treat the edge relations in a dependency tree as independent tokens in the input sequence. Then, we can present the dependency tree structure to the model by manipulating attention masks. With our method, the relation as an independent token can provide structural information to the text tokens and attend to the connected text tokens to gather contextual information about this relation. With the two-way interaction between the text and relation tokens, the model can learn a more feature-rich representation of each token and the whole sentence, including contextual and structural information.

We performed extensive experiments on the GLUE benchmark. The experiment results showed that the proposed framework can improve Transformer-based language models' performances in various downstream tasks. We also conducted experiments to show that the deeply integrated interaction between words and relations created by our proposed mechanism can lead to better performances than previous works.

Our contributions can be summarized as follows:

1. We proposed a novel way of presenting the structural information in a dependency tree by manipulating the attention masks of the Transformer, which we called the Dependency-aware Attention Masking (DAM) framework.
2. Since our method does not modify the architecture of the Transformer, it can be easily adapted to different tasks and be used with different Transformer-based pre-trained language models.
3. Experiments showed that the proposed DAM framework could further increase the performance of pre-trained language models on various sequence classification tasks.

The structure of the rest of the paper is as follows: In Section 2, we discuss some related works that incorporated dependency trees in NLP tasks. In Sec-



(a) G2GTr[7] (b) DT[5]

(c) DAM

Fig. 1: Simplified data flow diagram of the compared models and the proposed framework. The red arrows indicate interactions related to relation embeddings.

tion 3, we will introduce our proposed model framework in detail. In Section 4, we describe our experiments and present the results. Finally, we present our conclusions in Section 5.

2 Related Works

Several studies utilized GNNs to process the structural information embedded in the dependency tree. [16] uses Graph Convolution Network (GCN) on a pruned dependency tree to obtain word representations for relation extraction. [2] proposed an Attention-Guided GCN (AGGCN) module to use attention to "soft-prune" the dependency tree when processing the graph information. [3] employs a multi-layer Graph Attention Network (GAT) with LSTM cells between the layers for aspect-level sentiment classification. [13] also uses a multi-layer GAT to produce sentence representations from dependency trees and then applies it to document-level neural machine translation.

[15] design an aspect-aware attention mechanism to augment the GCN module that works alongside a bi-directional LSTM network for aspect-based sentiment analysis. [14] designed several ways of incorporating the dependency tree information encoded by a GCN with another encoder-decoder architecture for the opinion role labeling task. [4] employs a multi-layer GAT with layer aggregation on top of a Transformer encoder to further encode the sentence with

dependency information and applies it to the summarization task. [10] designs a bidirectional GCN and uses it beside a Transformer[11] encoder with a BiAffine module that allows the two modules to interact with each other in each layer.

Graph2Graph Transformer (G2GTr) [7] took inspiration from the relative position embedding from [9] and added additional relation embeddings in the self-attention block. When attending to a word that was connected with the current word on the dependency tree, the model would be informed about the dependency relation between the words. Dependency Transformer (DT) [5] took a different approach by adding a bias into the calculated self-attention score based on the relation type on the edge and the context of the connected words.

3 Methodology

3.1 Overview of the DAM Framework

Fig. 2 shows the overall workflow of the Dependency-aware Attention Masking (DAM) framework. Given a natural language sentence and the parsed dependency tree by some existing dependency parser. First, we transform the dependency tree to make better use of the edge labels. Next, we construct our input sequence and attention mask according to the transformed tree. Finally, we can use a Transformer encoder model to produce a set of output embeddings that corresponds to each position in the input sequence.

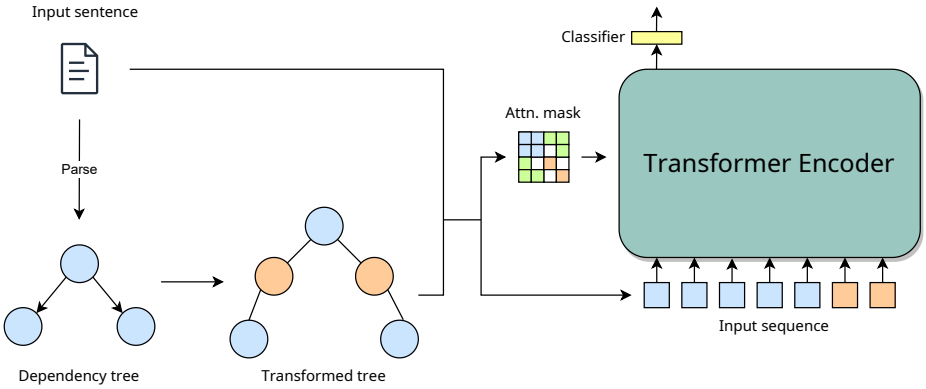


Fig. 2: Overview of the DAM framework.

3.2 Dependency Tree Transformation

As we have discussed in Section 1, both DT and G2GTr have the one-way interaction problem in which the interaction between the semantic information

in the words and the structural information in the relations only occurs in one direction. To better utilize the information carried by the relation type labels, we propose treating the edge itself as an independent unit instead of an abstract concept that connects two text tokens. In other words, we transform the edge labels of the original dependency tree into new nodes and construct a new type of tree.

Given a parsed dependency tree $G = (W, E, L)$, where W is the node set consisting of words in the sentence, E is the set of edges in the dependency tree, and L is the relation type labels of the edges. (w_i, l_j, w_z) is an edge connecting from w_i to w_z with relation type of l_j, where $w_i, w_z \in W, l_j \in L$ and $(w_i, l_j, w_z) \in E$. We perform a transformation process as follows. First, we treat the labels in the original tree as a new type of node and add them to the node set: $W' = W + L$. Then, for the edges in the original tree, we separate them into two edges that connect the label to its connected word nodes, respectively. For each word pair that was connected in the original tree, we connect the words to the label nodes instead of connecting them directly: $E' = \{(w_i, l_j),\ (w_z, l_j) \mid \forall\ (w_i, l_j, w_z) \in E\}$. We then define the transformed dependency tree as $G' = (W', E')$.

3.3 Input Representations

Building the Input Sequence Since the proposed transformed dependency tree has both word and relation nodes, we encode the two types of nodes into embedding sequences separately. The embedding sequences are combined later to form the final input sequence to the Transformer model.

For the word nodes, we use a tokenizer to turn the words into tokens while retaining their order in the original sentence. We insert a special token $[CLS]$ into the front of the sequence to mark the beginning of a sentence. The final output representation of the $[CLS]$ token is also used as the representation of the input sequence in sentence classification tasks. Another special token, $[SEP]$, is added to the end of the sequence to represent the end of the sentence. In tasks that involve multiple input sentences, such as the question-answering task, an additional $[SEP]$ token is inserted between the two sentences to separate them.

After tokenization, we use three types of embedding matrices to encode the different aspects of information in a token: token, segment, and position embeddings. The token embedding (Emb_t) encodes the semantic information of the respective token. The segment embedding (Emb_s) is helpful in tasks involving multiple input sentences as it can indicate which sentence the token belongs to in the input. The position embedding (Emb_p) represents the token's position in the entire input sequence.

For the relation nodes, we added new relation embedding (Emb_e) matrices for encoding the type of the relation labels. We treat the relation tokens as a new type of word in the input sequence and use embeddings of the same size as the word tokens to encode the structural information in the relation tokens. To make the model aware of the difference between the word tokens and the relation tokens, we add a relation type embedding vector $e_l \in R^d$ to the input embeddings of the relation tokens.

Finally, we construct the complete input sequence X by attaching the relation embedding sequences after the word embedding sequence. We provide an example of the constructed input sequence in Figure 3.

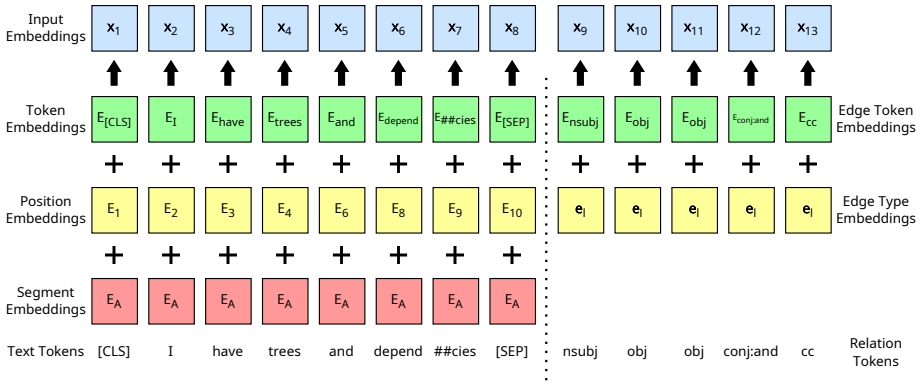


Fig. 3: An example input sequence of our model.

Attention Mask Construction Now, we have both words and relations as tokens in the input sequence. We could make the model aware of the structure of the transformed dependency tree by using the attention mask mechanism in Transformer. We design the attention mask construction process as follows:

We keep the bidirectional self-attention calculation between the word tokens, allowing the model to utilize the Transformer's self-attention mechanism to the most potential.

For between the word and the relation tokens, we only allow attention calculations when they are connected to each other in the transformed dependency tree. Since we are using the $[CLS]$ token as the aggregated representation of the entire sequence, the $[CLS]$ token should have the best view of all the information in the sequence. Therefore, we allow the $[CLS]$ token to attend to and be attended by all relation tokens. Also, if a word is separated into multiple subtokens by the tokenizer, we only treat the first subtoken as the representation of the word when constructing attention masks.

Lastly, for the edge tokens, we turned off the ability to attend to other edge tokens and only allowed the edge tokens to attend to themselves. This design is equivalent to adding self-loops to the dependency structure.

Our attention masking scheme can be summarized as follows: $A \in R^{n \times n}$ is the attention mask matrix, where n is the length of the input sequence. The

values in the matrix are defined as in Equation 1.

$$a_{ij} = \begin{cases} 1 & \text{if } i = 0 \vee j = 0, \\ 1 & \text{if } x_i, x_j \in T, \\ 1 & \text{if } x_i \in T \wedge x_j \in L \wedge (x_i, x_j) \in E', \\ 1 & \text{if } x_i, x_j \in L \wedge i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

3.4 Fine-tuning

With the constructed embedding sequence and attention mask, we can then obtain the output embeddings with a Transformer[11] model. The output embeddings of the final layer can then be used in different ways to adapt to various downstream tasks. One can connect the model to a decoder module to perform sequence-to-sequence tasks or use a classifier on each output embedding to do a token classification task. For sequence classification tasks, which we will focus on in the experiments, we use the output embedding of the first token ($[CLS]$) as the representation of the entire sequence. We use a simple linear classification layer $W^C \in R^{d \times k}$ followed by a softmax function to obtain the probability distribution P. Finally, we can train the entire model with the cross-entropy loss:

$$Loss = -\sum_{i=1}^{k} \hat{p_i} log(p_i) \tag{2}$$

where k is number of classes, p_i is the predicted probability for each class, and $\hat{p_i}$ is the ground truth.

4 Experiments

4.1 Dataset Descriptions

The General Language Understanding Evaluation (GLUE) benchmark[12] contains a collection of 9 natural language understanding tasks of different types, such as linguistic acceptability, sentiment analysis, semantic similarity, and textual entailment. The GLUE organizers also provide an online platform[3] that is open for public submission for evaluation and comparison across different models on their private test data. Descriptions of the rest of the datasets are listed in Table 1. Task type refers to the number of input sentences used in the task. Single means the model is tasked to determine the sentence's class given one sentence at a time. Pair means the model is given two input sentences at once and has to predict the correct class based on the semantic relation between the two sentences. The third column shows the number of samples provided in each dataset split. The rightmost column shows the respective metrics for evaluating the models' performances.

[3] https://gluebenchmark.com/

Table 1: Descripitions of each dataset in GLUE

Dataset	Task type	# of entries (train/dev/test)	Evaluation Metric
CoLA	Single	8,551/1,043/1,063	Matthew's corr.
STS-B	Pair	5,749/1,500/1,379	Pearson/Spearman corr.
MRPC	Pair	3,668/408/1,725	F1/Accuracy
RTE	Pair	2,490/277/3,000	Accuracy
SST-2	Single	67,349/872/1,821	Accuracy
QNLI	Pair	104,743/5,463/5,463	Accuracy
QQP	Pair	363,846/40,430/390,965	F1/Accuracy
MNLI	Pair	392,702/9,815/9,796 (matched) -/9,832/9,847 (mismatched)	Accuracy

4.2 Dependency Parser Settings

We use the Stanford CoreNLP toolkit[6] to parse the input sentences into dependency trees in all of our datasets. To increase the richness of the information a relation token possesses, we adopted the *enhanced++* dependency representation[8]. In the *enhanced++* representation, some parsed relations have a special relation quantificational modifier attached to its basic relation category. These modifiers can indicate the partitives or the prepositions, further describing the relationship between the connected words besides their basic categories. The fine-grained information from the modifiers is beneficial to discover the hidden information lying under dependency relations.

We treat each combination of a basic category and a quantificational modifier as a unique relation token and learn an embedding for each of them. Since the number of possible combinations of relation tokens is unpredictable, we filter out the *enhanced++* relations tags that appeared less than five times in our datasets, leaving 567 unique relation types in our vocabulary. We fall back to using its basic relation category for relations produced by the dependency parser but did not appear in our filtered vocabulary.

4.3 Performance Improvements to Language Models

We compare the performance improvements on the GLUE benchmark's test set. Since the labels in the test set were hidden from the public, we selected the best models based on the development set results to predict the answers and uploaded the results to GLUE's online platform for grading. We followed BERT's presentation format[1] in their original paper to show the results in Table 2.

Table 2: Fine-tuning results on the GLUE's test sets

Models	CoLA	STS-B	MRPC	RTE	SST-2	QNLI	QQP	MNLI-(m/mm)
BERT	52.1	**85.8**	**88.9**	66.4	**93.5**	90.5	71.2	84.6/83.4
BERT + DAM	**52.9**	85.4	87.2	**68.8**	93.2	**90.9**	**71.6**	**84.8/83.5**

4.4 Comparison with Other Methods

Table 3 shows the comparison results with other structure-aware Transformer models. The proposed DAM framework outperformed both DT and G2GTr on both tasks. Experiment results prove that the deeply integrated two-way interaction between the word and relation token introduced by the proposed scheme can help the model understand the dependency structure in the sentence better than the one-way methods.

Table 3: Comparison with other models on the single-sentence datasets.

Models	BERT	BERT+DT	BERT+G2GTr	BERT+DAM
CoLA	58.14	58.27	22.00	**58.44**
SST-2	92.32	92.54	90.94	**92.89**

5 Conclusions

In this study, we proposed an effective approach to enhance the structural awareness of the Transformer model. By changing the input sequence and the attention mask, we were able to represent the structural information in a dependency tree to the model besides the plain text sentence without changing the network architecture of the Transformer. Our approach creates a deeper interaction between the semantical and structural information bared in the input sequence compared to other methods. Our method is versatile and can work with most of the Transformer-based pre-trained language models. Experiments showed that our method could improve the performance of Transformer-based language models in various sentence classification tasks.

References

1. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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). pp. 4171–4186 (Jun 2019)
2. Guo, Z., Zhang, Y., Lu, W.: Attention guided graph convolutional networks for relation extraction. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 241–251 (Jul 2019)
3. Huang, B., Carley, K.: Syntax-aware aspect level sentiment classification with graph attention 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). pp. 5469–5477 (Nov 2019)

4. Jin, H., Wang, T., Wan, X.: Semsum: Semantic dependency guided neural abstractive summarization. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8026–8033 (2020)
5. Ma, J., Li, J., Liu, Y., Zhou, S., Li, X.: Integrating dependency tree into self-attention for sentence representation. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8137–8141 (2022)
6. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014)
7. Mohammadshahi, A., Henderson, J.: Graph-to-graph transformer for transition-based dependency parsing. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 3278–3289 (Nov 2020)
8. Schuster, S., Manning, C.D.: Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 2371–2378 (May 2016)
9. Shaw, P., Uszkoreit, J., Vaswani, A.: 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). pp. 464–468 (Jun 2018)
10. Tang, H., Ji, D., Li, C., Zhou, Q.: Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 6578–6588 (Jul 2020)
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. vol. 30 (2017)
12. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: 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. pp. 353–355 (Nov 2018)
13. Xu, M., Li, L., Wong, D.F., Liu, Q., Chao, L.S.: Document graph for neural machine translation. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 8435–8448 (Nov 2021)
14. Zhang, B., Zhang, Y., Wang, R., Li, Z., Zhang, M.: Syntax-aware opinion role labeling with dependency graph convolutional networks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 3249–3258 (Jul 2020)
15. Zhang, C., Li, Q., Song, D.: 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). pp. 4568–4578 (Nov 2019)
16. Zhang, Y., Qi, P., Manning, C.D.: Graph convolution over pruned dependency trees improves relation extraction. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2205–2215 (Oct-Nov 2018)