Tree Knowledge Distillation for Compressing Transformer-Based Language Models

Jiafei Xu, Jianmin Ji, and Defeng Xie

University Of Science And Technology Of China. 96, 230026 Hefei, China.
{xujaifei,ustdxdf}@mail.ustc.cn
jianmin@ustc.edu.cn

Abstract. Knowledge distillation has emerged as a promising technique for compressing neural language models. However, most knowledge distillation methods focus on extracting the "knowledge" from a teacher network to guide the training of a student network, ignoring the "requirements" of the student. In this paper, we introduce Tree Knowledge Distillation for Transformer-based teacher and student models, which allows student to actively extract its "requirements" via a tree of tokens. In specific, we first choose the `[CLS]` token at the output layer of Transformer in student as the root of the tree. We choose tokens with the highest values in the row for `[CLS]` of the attention feature map at the second last layer as the children of the root. Then we choose children of these nodes in their corresponding rows of the attention feature map at the next layer, respectively. Later, we connect layers of Transformer in student to corresponding layers in teacher by skipping every t layer. At last, we improve the loss function by adding the summed mean squared errors between the embeddings of the tokens in the tree. The experiments show that tree knowledge distillation achieves competitive performance for compressing BERT among other knowledge distillation methods in GLUE benchmark.

Keywords: Tree Knowledge Distillation, Transformer, Multi-Head Attention.

1 Introduction

Pre-trained neural language models based on Transformer networks [15], like BERT [3], RoBERTa [10], XLNet [19], and GPT-3 [1], have achieved remarkable improvements on various natural language processing (NLP) tasks, such as question answering [7] and sentiment classification [12].

These large-scale language models require many parameters to obtain good performance. For instance, the BERT-base model has 12 layers and 110 million parameters, and the GPT-3 model has 175 billion parameters. It is challenging to deploy these language models to real-time applications in environments with limited computational resources. Recently, knowledge distillation [5, 4] has emerged as a promising technique to compress large-scale language models for

these applications. Patient Knowledge Distillation (PKD) [13] has been used to compress the 12-layer BERT model to the 6-layer and 3-layer models with little sacrifice on the performance.

Knowledge distillation is a general technique for guiding the training of a "student" neural network by capturing and transferring the "knowledge" of a pre-trained "teacher" network. The distillation loss is added to encourage the student to mimic some aspects of the teacher. When the teacher network is a Transformer-based model, the distillation loss added by vanilla knowledge distillation [5] is to encourage the student to mimic the output of the `[CLS]` token at the output layer of the teacher. On the other hand, it has shown that features of intermediate layers learned by the teacher can also be used to improve the training process and final performance of the student [4]. For a Transformer-based model, such features can be considered as the embeddings of corresponding tokens at the Transformer's intermediate layer. The distillation loss can then be improved to minimize the difference of these embeddings for corresponding tokens between the teacher and the student. However, it is inefficient to consider all tokens at each layer of the student. In PKD [13], the `[CLS]` token is only considered at each layer to compute the distillation loss, which has shown a promising improvement for compressing the BERT model.

Above knowledge distillation methods focus on extracting the teacher's knowledge to the student while ignoring the student's "requirements". In this paper, we further extend the idea by considering a tree of tokens for corresponding layers to allow the student to extract its requirements from the teacher actively. Following the idea, we introduce Tree Knowledge Distillation (TKD) for compressing Transformer-based language models. In specific, we first choose the `[CLS]` token at the output layer of the Transformer in the student as the root of the tree. We choose tokens with the highest values in the row for `[CLS]` of the attention feature map at the second last layer as the children of the root. Then we choose children of these nodes in their corresponding rows of the attention feature map at the next layer, respectively. Later, we connect layers of the Transformer in the student to corresponding layers of the Transformer in the teacher by skipping every t layer, when the number of layers in the teacher is larger than the student number. Finally, we improve the loss function of knowledge distillation by adding the summed mean squared errors between the embeddings of the tokens in the tree for corresponding layers of the teacher and the student.

Notice that, the tree of tokens is incrementally constructed by choosing tokens with the highest values in corresponding rows of attention feature maps at each layer of the student. In other words, the tree consists of tokens that are most interested in the student at the current training stage. The distillation loss of TKD encourages the student to mimic the embeddings of these tokens in the teacher, which allows the student to specify its requirements, i.e., the tree of tokens, and actively learn the required knowledge, i.e., the embeddings of corresponding tokens in the layers, from the teacher.

We evaluate TKD on multiple NLP tasks in General Language Understanding Evaluation (GLUE) benchmark [16]. The experiments show that TKD achieves

competitive performance for compressing BERT, among other knowledge distillation methods in GLUE. Note that, more tokens are considered in TKD than PKD. However, we observe little increase in the computational cost.

The main contributions of the paper are summarized as follows:

- We introduce Tree Knowledge Distillation (TKD) that allows a Transformer-based student network to actively extract knowledge from a pre-trained Transformer-based language model to guide its training.
- We propose to improve the distillation loss to minimize the difference of the embeddings for specific tokens at intermediate layers between the teacher and the student, where the student chooses the tokens following a tree structure.
- We implement TKD to compress the 12-layer BERT model to a 3-layer model. The experiments show that TKD achieves competitive performance for compressing BERT, among other knowledge distillation methods, on multiple NLP tasks in GLUE benchmark. Moreover, we compared the tree-shaped embedding selection rule with some other rules, and the results show that the performance of the tree-shaped rule is in the first echelon, and it can get better results when combined with other rules.

The rest of the paper is organized as follows. The related work is presented in the next section. Then, vanilla knowledge distillation, and the multi-head attention mechanism in Transformer are reviewed. Later, the details of TKD and the experimental results are proposed. At last, we conclude the paper.

2 Related Work

Knowledge distillation methods focus on extracting the transferring knowledge from a teacher network to guide a student network's training. In vanilla knowledge distillation [5], the teacher's softened class scores are considered the transferring knowledge, and the distillation loss is to minimize the difference of the scores between the teacher and the student. Later, the teacher's intermediate representations are also used to improve the student's training and final performance. For instance, the activations [11], neurons [6], or features [21] of intermediate layers can be considered as the knowledge. Moreover, the relationships between different activations [20], neurons [8], or pairs of samples [14] can also be used as the knowledge. Furthermore, the connections between the parameters of different layers in the teacher can also guide the training of the student [9].

Knowledge distillation approaches, like PKD [13], have been applied to compress large-scale language models for real-time applications in environments with limited computational resources. Unlike these approaches, we introduce TKD that allows a Transformer-based student network to actively extract knowledge from a pre-trained Transformer-based language model to guide its training.

3 Preliminaries

3.1 Knowledge Distillation

We first review the distillation loss in vanilla knowledge distillation [5], which is to encourage the student to mimic the output of the teacher. In specific, a cross entropy loss between the outputs of the student and the teacher is defined as,

$$L_{DS} = -\sum_{i=1}^{N}\sum_{j=1}^{B}\left[P(y_i = j \mid x_i;\, \theta^t)\ \log P(y_i = j \mid x_i;\, \theta^s)\right], \tag{1}$$

where N is the number of training samples, B is the number of categories of labels, y_i is the output of the model for input x_i, θ^t denotes the parameters of the teacher model, and θ^s denotes the parameters of the student.

Moreover, the student model should also minimize the training loss. In specific,

$$L_{CE} = -\sum_{i=1}^{N}\sum_{j=1}^{B}\left[\mathbf{1}(y_i = j)\ logP(y_i = j \mid x_i;\, \theta^s)\right], \tag{2}$$

where $\mathbf{1}(y_i = j)$ returns 1 if y_i is j and 0 otherwise.

At last, the overall loss function of the student network incorporates both knowledge distillation and knowledge loss. In specific,

$$L_{KD} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE}, \tag{3}$$

where α is a hyper-parameter that controls the weight of the distillation loss.

3.2 Multi-Head Attention

Multi-head attention [15] is a type of dot-product attention, which maps a query and a set of key-value pairs to an output in the Transformer structure.

The query, the key, and the value are all vectors computed from the input. For a batch of sentences with the batch size b, attention first uses different learned linear projection functions to get the query matrix Q, the key matrix K, and the value matrix V. Then, the output is calculated by the following formula,

$$Attention(Q, K, V) = softmax(\frac{Q\,K^T}{\sqrt{d_k}})V, \tag{4}$$

where d_k is the size of the first dimension of the K. The dot product of Q and K will grow as the dimension of the key grows, so it needs to be divided by $\sqrt{d_k}$ to neutralize this.

In order to speed up the calculation and improve the accuracy, attention uses multiple heads to calculate the results in parallel. First, they divide each input with d_{model} dimensions to h pieces and use h projections to make h sets of the query, key, and value. Each head is responsible for processing one set, and different heads can compute parallelly. Finally, outputs of each head are concatenated

and projected again, resulting in the final values. The whole procession can be denoted as,

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)\, W^O, \tag{5}$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, W_i^Q, W_i^K, and W_i^V denote the linear projections of $head_i$, and W^O is the final projection.

4 Tree Knowledge Distillation

In this section, we introduce Tree Knowledge Distillation (TKD) for compressing Transformer-based language models.

TKD uses features of intermediate layers learned by the teacher to improve the final performance of the student. In TKD, such features are considered as the embeddings of corresponding tokens at layers of the Transformer model. In specific, given an input sentence x_i, the embedding of the token p at jth level of the teacher (resp. the kth level of the student) is denoted as $h^t_{i,j,p}$ (resp. $h^s_{i,k,p}$). The function I_{pt} maps a layer k in the student to a layer j in the teacher. We define $I_{pt}(k) = 4k$ in our experiments on compressing a 12-layer BERT to a 3-layer model. It means that the student model selects one Transformer from the teacher for learning every 4 layer. Each Transformer layer of the student has one corresponding layer of Transformer from the teacher to learn.

As discussed in the above sections, TKD allows the student to actively extract knowledge from the teacher via a tree of tokens, which is constructed as follows. We first choose the `[CLS]` token at the output layer of the student as the root of the tree. We choose tokens with n highest values in the row for `[CLS]` of the attention feature map at the second last layer as the children of the root. Then we choose children of these nodes in their corresponding rows of the attention feature map at the next layer, respectively. Specifically, we use td(h,k) to represent the mark selected in the tree of the h-th head of the student at the k-th level. Notice that, in order to restrict the size of the tree, the maximum number of children from a node is required to be a fixed number m. We set $m = 2$ in our experiments on compressing a 12-layer BERT to a 3-layer model. An example of such a tree of tokens is illustrated in Figure 1.

Then TKD adds the summed mean squared errors between the embeddings of the tokens in the tree at corresponding layers of the teacher and the student. In specific,

$$L_{TD} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{p \in td(h,k)} \left\| \frac{h^s_{i,k,h,p}}{\|h^s_{i,k,h,p}\|_2} - \frac{h^t_{i,I_{pt}(k),h,p}}{\|h^t_{i,I_{pt}(k),h,p}\|_2} \right\|_2^2. \tag{6}$$

Note that there are 12 trees in the process of TKD compressing the BERT-base, and each tree only processes the 64-dimensional tensor it is responsible for The 12 trees handle a total of 768-dimensional tensor, which is exactly the embedding size of BERT-base. In function (6), $h^s_{i,k,h,p}$ is a 64-dimensional tensor.

In addition, we let the `[CLS]` embedding of each layer of the student model learn from the [CLS] embedding of the corresponding Transformer layer in the teacher. This means that the loss function adds the mean square error of the sum, as shown in the following function:

$$L_{PT} = \sum_{i=1}^{N} \sum_{k=1}^{K} \left\| \frac{h^s_{i,k,0}}{\|h^s_{i,k,0}\|_2} - \frac{h^t_{i,I_{pt}(k),0}}{\|h^t_{i,I_{pt}(k),0}\|_2} \right\|_2^2 \tag{7}$$

Finally, the overall loss function of TKD is written as:

$$L_{TKD} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE} + \gamma\, L_{TD} + \beta\, L_{PT}, \tag{8}$$

where γ denotes the importance of the loss L_{TD} and β denotes the importance of the loss L_{PT}. The entire TKD calculation process is shown in Algorithm 1 1

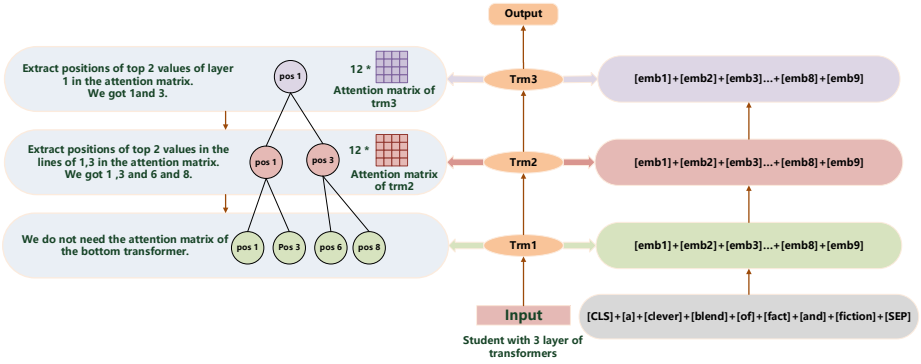

Fig. 1. The process of selecting nodes in the student model of the 3-layer Transformer.

4.1 Datasets

5 Experiments

In this section, we implement TKD for compressing the 12-layer BERT model to a 3-layer model. We evaluate the compressed model with multiple NLP tasks in the GLUE benchmark. The experiments show that TKD achieves competitive performance for compressing BERT, among other knowledge distillation methods, with little increase of the computational cost. We also show that TKD can be fruitfully combined with other existing knowledge distillation methods to achieve better performance.

We evaluate TKD on four NLP tasks in General Language Understanding Evaluation (GLUE) benchmark [16]. GLUE consists of nine sentence or sentence-pair NLP tasks built on well-established existing datasets. These datasets are

Algorithm 1 Tree Knowledge Distillation

Input: Transformer models for the teacher and the student **Output**: The trained student model via TKD

Fine-tune the teacher model on the target dataset
Set the checkpoint as the epoch to start TKD
for every input sentence x_i in the dataset **do**
  Compute the KD loss L_{KD} by Equation (3)
  Compute the PT loss L_{PT} by Equation (7)
  if epoch > checkpoint **then**
    Set the `[CLS]` token at the output layer of the student as the root of the tree
    for every kth layer of the student from K to 1 **do**
      for every hth head of the Transformer layer in student **do**
        for each token p in the previous layer of the tree corresponding to h head **do**
          Choose m tokens with the highest values in the row for p of the attention feature map at the kth layer
          Append these tokens to the tree
          Add these tokens to $td(k)$
        end for
      end for
    end for
    Compute the loss L_{TD} by Equation (6)
    Compute the TKD loss L_{TKD} by Equation (8)
    Optimize the student network via the TKD loss
  end if
end for

selected to cover a diverse range of dataset sizes, text genres, and degrees of difficulty. We focus on four of them, i.e., Multi-Genre Natural Language Inference (MNLI) [18] which contains MNLI-matched and MNLI-mismatched, Stanford Sentiment Treebank (SST-2) [12], Quora Question Pairs (QQP) [2] and Recognizing Textual Entailment (RTE). Notice that, MNLI and RTE concern the Natural Language Inference task, SST-2 concerns the Sentence Sentiment Classification task, and QQP concerns the Question Answering task.

5.1 Training Details

Notice that, NLP tasks in the above datasets can be considered as classification problems. Then both models of the student and the teacher can be constructed as a Transformer model with the softmax layer for the output. We convert the input sentence S into the sentence `[CLS]` + S + `[SEP]` and the pair of sentences S_1, S_2 into the sentence `[CLS]` + S_1 + `[SEP]` + S_2 + `[SEP]`, following the preprocess in [3] for BERT. Due to the limitation of computational resources, we set the upper bound of the length for input sentences to be 150, while truncating longer sentences.

We use BERT-base with 12 layers to construct our teacher model, whose parameters come from the bert-base-uncased parameter provided by hugging-

face[1]. The Tokenizer is also obtained from huggingface. Note that, we fine-tune the teacher for corresponding datasets before applying TKD. We found that parameters in the fine-tuned teacher model can give the student a better starting point than the original parameters in BERT. Our student model follows a 3-layer Transformer structure, whose initial parameters are obtained from parameters at the 4th, 8th, and 12th layer of the Transformer in the teacher.

In addition to using the tree path to select embedding for learning, we also chose other rules to select embedding for comparison, including RAND, ALL, Teacher-TKD, vanilla knowledge distillation (denoted as KD), and PKD. To get the best fine-tuned teacher, we choose the learning rate from {1e-5, 2e-5, 5e-5}, and use the one with the best performance. To obtain good hyper-parameters for the student, we set the temperature as $\{5,\ 10,\ 20\}$, the KD weight coefficient α as $\{0.2,\ 0.5,\ 0.7, 0.9\}$, the TD weight γ as $\{10,\ 50,\ 100,\ 500,\ 1000\}$ and the pt weight β as $\{100,\ 500,\ 1000,\ 5000, 10000\}$. Furthermore, we set the maximum training epoch to be 30, and conduct 5 experiments on each group of parameters to find the trained student models with the best performance.

5.2 Experimental Results

We submitted the predictions of corresponding models to GLUE's official evaluation server. Table 1 summarizes the results of trained models. In specific, "BERT_{12} (Google)" denotes the teacher model without fine-tune. "BERT_{12} (Teacher)" denotes our fine-tuned teacher model. "BERT_3" denotes our fine-tuned student model without using Knowledge Distillation. "BERT_3-KD" denotes the trained student model via the vanilla Knowledge Distillation approach. "BERT_3-PKD" denotes the trained student model via the Patient Knowledge Distillation approach. "BERT_3-TKD" denotes the trained student model via the Tree knowledge distillation approach. "$\text{BERT}_(3)$-ALL" denotes that each layer in the student model learns all the embeddings of the corresponding layer of the teacher model. "BERT_3-RAND" denotes that randomly select a fixed number of N nodes to learn for the part corresponding to each head of each layer of the student model. "$\text{BERT}_(3)$-Teacher-TKD" means that the attention matrix of the teacher is used to replace the attention-matrix of the student in the TKD algorithm. "$\text{BERT}_(3)$-T-S-TKD" means that the TKD algorithm was run twice with the attention matrix of the teacher model and the attention matrix of the student. "$\text{BERT}_(3)$-ALL-T-TKD" means that each layer in the student model learns all the embeddings of the corresponding layer and the Teacher-TKD chosen embeddings of the teacher model. "BERT_3-ALL-TKD" denotes each layer in the student model learns all the embeddings of the corresponding layer and the TKD chosen embeddings of the teacher model. "BERT_3-ALL-T-S-TKD" denotes each layer in the student model learns all the embeddings of the corresponding layer, the TKD chosen embeddings, and the Teacher-TKD chosen embeddings of the teacher model.

[1] https://huggingface.co

Table 1. Different student models on GLUE benchmark.

Model	SST-2 (67k)	MNLI-m (393k)	MNLI-mm (393k)	QQP (364k)	RTE(2.5k)
$BERT_{12}$ (Google)	93.5	84.6	83.4	71.2/89.2	66.9
$BERT_{12}$ (Teacher)	94.3	83.7	82.8	70.9/89.0	69.1
$BERT_3$-DistilBERT	91.3	82.2	-	-/88.5	59.9
$BERT_3$	86.4	74.8	74.3	65.8/86.9	55.2
$BERT_3$-KD	86.9	75.4	74.8	67.3/87.6	56.2
$BERT_3$-PKD	87.5	76.7	76.3	68.1/87.8	58.2
$BERT_3$-TKD	91.5	81.6	79.3	69.1/88.6	59.5
$BERT_3$-ALL	91.2	81.8	79.3	68.9/88.4	59.5
$BERT_3$-RAND	89.8	77.1	76.5	67.2/87.4	58.3
$BERT_3$-Teacher-TKD	90.5	78.5	77.0	69.6/88.6	57.6
$BERT_3$-T-S-TKD	91.3	81.6	79.1	68.9/88.4	59.3
$BERT_3$-ALL-T-TKD	91.1	80.9	78.3	68.9/88.2	58.9
$BERT_3$-ALL-TKD	**91.9**	**82.9**	**79.6**	**70.2/88.7**	**60.6**
$BERT_3$-ALL-T-S-TKD	91.6	82.6	79.3	69.9/88.7	60.1

In Table 1, besides the mixed approach, $BERT_3$-TKD achieves best results in all datasets, and $BERT_3$–ALL-TKD outperforms all other approaches. The experiment results show that TKD achieves competitive performance for compressing BERT, among other knowledge distillation methods in GLUE. The comparison between the results of $BERT_3$-TKD, $BERT_3$-Teacher-TKD and $BERT_3$-RAND shows that: When the student model actively choose their focus to study, it can achieve better results than passive learning. Letting students take the initiative to learn knowledge is a direction worth exploring. The experimental results of $BERT_3$-TKD, $BERT_3$-Teacher-ALL and $BERT_3$-PKD show that TKD can allow student models to learn a wealth of knowledge and prove the superiority of the tree structure. The results show that letting students learn the knowledge they care about can achieve better results than blindly learning all the knowledge.

In specific, on two of the largest datasets in our experiment, i.e., MNLI-m and MNLI-mm, $BERT_3$-TKD improves the performance of $BERT_3$ by 6.8% and 5.0% of the accuracy, improves $BERT_3$-KD by 6.2% and 4.5%, and improves $BERT_3$-PKD by 4.9% and 3%. For SST-2, $BERT_3$-TKD improves $BERT_3$ by 5.1%, improves $BERT_3$-KD by 4.6%, and improves $BERT_3$-PKD by 4%. For QQP, $BERT_3$-TKD improves $BERT_3$ by 3.3%, improves $BERT_3$-KD by 1.8%, and improves $BERT_3$-PKD by 1%. For RTE, $BERT_3$-TKD improves $BERT_3$ by 4.9%, improves $BERT_3$-KD by 3.9%, and improves $BERT_3$-PKD by 1.9%. Experiments show that the larger the dataset, the better the effect of TKD.

5.3 Ablation Study

The knowledge distillation has made independent improvements for compressing language models. It is clear that knowledge can be transferred from teacher to

student by learning from embeddings. However, it is not yet clear what strategy to choose embeddings for learning will enable students to achieve the best performance. Here we examine the combination of some other rules for choosing embeddings and TKD. Note that, KD is already considered in either TKD and other rules. The ALL-TKD can be considered as choosing both TKD embeddings and all embeddings to learn. In specific, the loss function for the ALL-TKD approach is defined as:

$$L_{ALL\text{-}TKD} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE} + \beta\, L_{PD} + \gamma\, L_{TD} + \eta\, L_{ALL}. \tag{9}$$

where L_{ALL} is defined as the following function:

$$L_{ALL} = \sum_{i=1}^{N} \sum_{k=1}^{K} \left\| \frac{h_{i,k}^{s}}{\|h_{i,k}^{s}\|_2} - \frac{h_{i,I_{pt}(k)}^{t}}{\|h_{i,I_{pt}(k)}^{t}\|_2} \right\|_2^2 \tag{10}$$

The experimental results are shown in Table 1, which show that TKD added with the strategy of learning all the embeddings can get the best result. We found that when the size of the dataset is larger, better improvement of performance would be observed. In specific, compared with BERT_3-TKD, BERT_3-ALL-TKD improves the performance by 0.4% on SST-2, 1.3% on MNLI-m, 0.3% on MNLI-mm, and 0% on QQP. We also found that no matter it is combined with TKD or ALL, Teacher-TKD can not get better results.

5.4 Discussion

Experimental results in Table 1 show that TKD and the ALL-TKD approach provide better improvements for the performance of the fine-tuned student model on datasets with the larger size. This observation implies that TKD and the ALL-TKD approach are more suitable for the cases that the teacher model is well-trained and the smaller student model has troubles to be trained well for a large number of training samples.

Different from other knowledge distillation methods, TKD allows the student to actively acquire knowledge from the teacher due to its own interests. This is very interesting, because this learning method corresponds to the focused learning method in human learning process. Experiments show that TKD mechanism can further improve the performance of other knowledge distillation methods while introducing little increase in the computational cost.

On the other hand, language models can be regarded as knowledge graphs [17]. In the knowledge distillation procession, the student intends to learn the complete knowledge graph w.r.t the teacher's language model. Due to the limits of abilities and resources, knowledge distillation only captures and transfers partial knowledge of the teacher, which respects some knowledge in the knowledge graph for the teacher. Intuitively, it is more helpful if such knowledge form relative paths towards the final result, i.e., entities correspond to the `[CLS]` token at the output layer of the student. Vanilla knowledge distillation can be regarded as equivalent to encourage the student to mimic the embedding output of the

[CLS] token at the output layer of the teacher, which only transfers knowledge of entities for the result. PKD encourages the student to mimic the embeddings of [CLS] at each layer of the student, where the corresponding knowledge are useful to construct the final result in the knowledge graph. Clearly, knowledge w.r.t. other tokens is also useful for the final result. In TKD, we further extend the idea by conferring more weights to entities that correspond to certain tokens in a tree structure, that are most interested in the student at the current training stage. Intuitively, the sequences of these entities form relative paths towards the construction of the final result.

6 Conclusion

In this paper, we introduce Tree Knowledge Distillation (TKD) that allows a Transformer-based student network to actively extract knowledge from a pre-trained Transformer-based language model to guide its training. Different from other knowledge distillation methods, TKD allows the student to actively acquire knowledge from the teacher due to its interests. TKD improves the distillation loss to minimize the difference of the embeddings for chosen tokens at intermediate layers between the teacher and the student, where the student chooses the tokens with the most interests in a tree structure. We implement TKD to compress the 12-layer BERT model to a 3-layer model. The experiments show that TKD achieves competitive performance for compressing BERT, among other knowledge distillation methods, on multiple NLP tasks in the GLUE benchmark. Moreover, experimental results also show that TKD can be fruitfully combined with some other embedding choosing strategy methods to achieve better performance.

In the future, we intend to further explore TKD from the point of view of the relations between language models and knowledge graphs. We would be committed to providing a theoretical basis for the mechanic analysis of knowledge distillation.

References

1. Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
2. Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao.: Quora question pairs (2018).
3. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2019)
4. Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao.: Knowledge distillation: A survey. arXiv preprint arXiv:2006.05525 (2020)
5. Geoffrey Hinton, Oriol Vinyals, and Jeff Dean.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)

6. Zehao Huang and Naiyan Wang.: Like what you like: Knowledge distill via neuron selectivity transfer. arXiv preprint arXiv:1707.01219 (2017)
7. Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017)
8. Seunghyun Lee and Byung Cheol Song.: Graph-based Knowledge Distillation by Multi-head Attention Network. arXiv preprint arXiv:1907.02226 (2019)
9. Junjie Liu, Dongchao Wen, Hongxing Gao, Wei Tao, Tse-Wei Chen, Kinya Osa, and Masami Kato. Knowledge Representing: Efficient, Sparse Representation of Prior Knowledge for Knowledge Distillation. In*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 638-646 (2019)
10. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692 (2019)
11. Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets. In *Proceedings of the 3rd International Conference on Learning Representations, (ICLR-2015)* (2015)
12. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* pages 1631–1642 (2013)
13. Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. 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 4314–4323 (2019)
14. Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision* pages 1365–1374 (2019)
15. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems* pages 5998–6008 (2017)
16. Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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 (2018)
17. Chenguang Wang, Xiao Liu, and Dawn Song.: Language Models are Open Knowledge Graphs. arXiv preprint arXiv:2010.11967 (2020)
18. Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* pages 1112–1122 (2018)
19. Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems* pages 5753–5763 (2019)

20. Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pages 4133–4141 (2017)
21. Sergey Zagoruyko and Nikos Komodakis.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928 (2016)