Tree Knowledge Distillation for Compressing Transformer-Based Language Models

Anonymous ACL submission

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 layers. 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.

1 Introduction

Pre-trained neural language models based on Transformer networks Vaswani et al. [2017], like BERT Devlin et al. [2019], RoBERTa Liu et al. [2019b], XLNet Yang et al. [2019], and GPT-3 Brown et al. [2020], have achieved remarkable improvements on various natural language processing (NLP) tasks, such as question answering Lai et al. [2017] and sentiment classification Socher et al. [2013].

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 Hinton et al. [2015]; Gou et al. [2020] has emerged as a promising technique to compress large-scale language models for these applications. Patient Knowledge Distillation (PKD) Sun et al. [2019] 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 Hinton et al. [2015] 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 Gou et al. [2020]. 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 Sun et al. [2019], 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 layers, 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.

On the other hand, [Wang et al., 2020] shows that a language model can be considered as an open knowledge graph and the process of the model training can be considered as the process of optimizing the knowledge graph. Then TKD for a language model can be viewed as a way of optimizing the knowledge graph for the student by conferring more weights to entities that correspond to tokens in the tree. Intuitively, the sequences of these entities form relative paths towards the final result, i.e., entities correspond to the `[CLS]` token at the output layer. Then by constructing the tree of tokens in TKD, we specify paths of entities that are considered to be more critical for the final result from the view of the student.

We evaluate TKD on multiple NLP tasks in General Language Understanding Evaluation (GLUE) benchmark Wang et al. [2018]. 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. Moreover, we show that TKD can be fruitfully combined with other existing knowledge distillation methods to achieve better performance.

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, experimental results also show that TKD can be fruitfully combined with other existing knowledge distillation methods to achieve better performance.

The rest of the paper is organized as follows. The related work is presented in the next section. Then, vanilla knowledge distillation, PKD, 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 Hinton et al. [2015], 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 Romero et al. [2015], neurons Huang and Wang [2017], or features Zagoruyko and Komodakis [2016] of intermediate layers can be considered as the knowledge. Moreover, the relationships between different activations Yim et al. [2017], neurons Lee and Song [2019], or pairs of samples Tung and Mori [2019] 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 Liu et al. [2019a].

Knowledge distillation approaches, like PKD Sun et al. [2019], 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 Hinton et al. [2015], 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,

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

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,

$$L_{KD} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE}, \tag{2}$$

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

3.2 Patient Knowledge Distillation

Patient Knowledge Distillation (PKD) Sun et al. [2019] focuses on compressing the 12-layer BERT model to a 3-layer model, where features of intermediate layers learned by the teacher are also used to improve the training process and final performance of the student.

In PKD, such features are considered as the embeddings of the `[CLS]` token at corresponding layers of the Transformer model. In specific, given an input sentence x_i, the output of the 12-layer BERT model (resp. the 3-layer model) is specified by the embedding of the `[CLS]` token at the output layer. The embedding of the `[CLS]` token at the jth level of the teacher (resp. the kth level of the student) is denoted as $h^t_{i,j}$ (resp. $h^s_{i,k}$) for $j \in \{1, \ldots, 12\}$ (resp. $k \in \{1, 2, 3\}$).

Note that the number of layers in the teacher is larger than the number in student. Then PKD defines a function I_{pt} that maps a layer k in the student to a layer j in the teacher, i.e., $I_{pt}(k) = j$. A simple way for such mapping is to connect layers in the student to corresponding layers in the teacher by skipping every t layers, i.e., $I_{pt}(k) = k\,t$. As shown in Figure 1, in our case, $t = 4$, $I_{pt}(1) = 4$, $I_{pt}(2) = 8$, and $I_{pt}(3) = 12$.

Then PKD adds the summed mean squared errors between the embeddings of the `[CLS]` token at corresponding layers of the teacher and the student. In specific,

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

where N is the number of training samples and K is the number of layers in the Transformer of the student.

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

$$L_{CE} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE} + \beta\, L_{PT},$$

where β denotes the importance of the loss between the features of the student and the teacher.

3.3 Multi-Head Attention

Multi-head attention Vaswani et al. [2017] 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

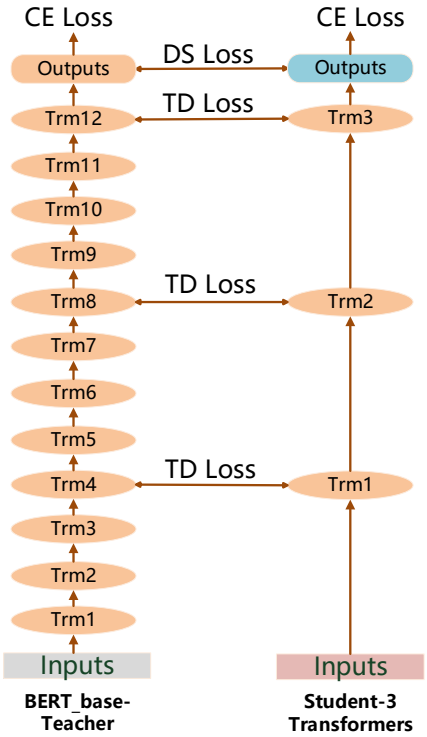

Figure 1: The Model architecture of PKD for compressing the 12-layer BERT model to a 3-layer model, where the student network learns the teacher's outputs in every 4 layers.

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,

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

where d_k is the dimension of 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,

$$\mathit{MultiHead}(Q, K, V) = \mathit{Concat}(\mathit{head}_1, \ldots, \mathit{head}_h)\, W^O,$$

where $\mathit{head}_i = \mathit{Attention}(QW_i^Q,\, KW_i^K, VW_i^V)$, 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 also uses features of intermediate layers learned by the teacher to improve the training process and 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. $h^s_{i,k,p}$).

Similar to the discussion in PKD, the function I_{pt} needs to be specified to map a layer k in the student to a layer j in the teacher. We also define $I_{pt}(k) = 4k$ in our experiments on compressing a 12-layer BERT to a 3-layer model.

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 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. In specific, we use $td(k)$ to denote the chosen tokens in the tree at the kth level of the student. 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 2.

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

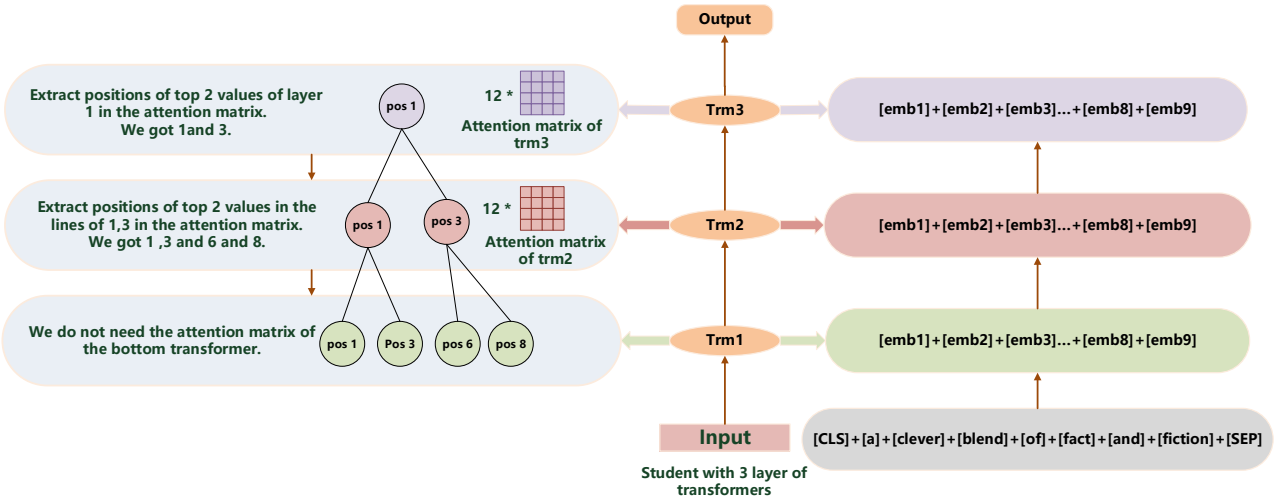

Figure 2: Constructing the tree of tokens for the Transform of the student in TKD.

student. In specific,

$$L_{TD} = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{p \in td(k)} \left\| \frac{h^s_{i,k,p}}{\|h^s_{i,k,p}\|_2} - \frac{h^t_{i,I_{pt}(k),p}}{\|h^t_{i,I_{pt}(k),p}\|_2} \right\|_2^2. \quad (3)$$

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

$$L_{TKD} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE} + \gamma\, L_{TD}, \quad (4)$$

where γ denotes the importance of the loss L_{TD}.

Algorithm 1 specifies the whole procedure of TKD.

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 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.

5.1 Datasets

We evaluate TKD on multiple NLP tasks in General Language Understanding Evaluation (GLUE) benchmark Wang et al. [2018]. GLUE consists of nine sentence or sentence-pair NLP tasks built on well-established existing datasets. These datasets are selected to cover a diverse range of dataset sizes, text genres, and degrees of the difficulty. We focus on five of them, i.e., Multi-Genre Natural Language Inference (MNLI) Williams et al. [2018], Recognizing Textual Entailment (RTE), Stanford Sentiment Treebank (SST-2) Socher et al. [2013], Quora Question Pairs (QQP) Chen et al. [2018], and Microsoft Research Paraphrase Corpus (MRPC) Dolan and Brockett [2005]. Notice that, MNLI and RTE concern the Natural Language Inference task, SST-2 concerns the Sentence Sentiment Classification task, QQP concerns the Question Answering task, and MPRC concerns the Paraphrase Similarity Matching task.

In specific, MNLI contains about 400k sentence pairs and corresponding labels. There are three types of labels, i.e., entailment, neutral, and contradiction, which represent the logical relationships between the two sentences. RTE is similar to MNLI. However, the volume of RTE is much smaller than that of MNLI, which contains 2.5k pieces of data. SST-2 is a sentence sentiment analysis dataset containing 67k pieces of data. Sentences in SST-2 are film reviews from the IMDB website, which correspond to scores between 0 to 1. The label is either positive or negative according to the scores. QQP obtains question-pairs from the website Quora, and the task is to determine whether the two questions are duplicated. Finally, MPRC contains 3.7k sentence pairs. The model needs to distinguish whether the two sentences are semantically equivalent, and the label is either yes or no. Among these tasks, QQP and MNLI are the two largest data sets, while MRPC and RTE are very small.

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 (2)
  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 each token p in the previous layer of the tree **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
    Compute the loss L_{TD} by Equation (3)
    Compute the TKD loss L_{TKD} by Equation (4)
    Optimize the student network via the TKD loss
  end if
end for

In specific, MNLI contains about 400k sentence pairs and corresponding labels. There are three types of labels, i.e., entailment, neutral, and contradiction, which represent the logical relationships between the two sentences. RTE is similar to MNLI. However, the volume of RTE is much smaller than that of MNLI, which contains 2.5k pieces of data. SST-2 is a sentence sentiment analysis dataset containing 67k pieces of data. Sentences in SST-2 are film reviews from the IMDB website, which correspond to scores between 0 to 1. The label is either positive or negative according to whether the score is greater than 0.5. QQP obtains labelled corpus of 364k question-pairs from the website Quora, and the task is to determine whether the two questions are duplicated. Finally, MPRC contains 3.7k sentence pairs. The model needs to distinguish whether the two sentences are semantically equivalent, and the label is either yes or no. Among these datasets, QQP, MNLI are the two largest datasets, while MRPC and RTE are small.

5.2 Training Details

Notice that, NLP tasks in 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 Devlin et al. [2019] 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 huggingface[1]. In addition, 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.

We compared the performance of the trained student via vanilla knowledge distillation (denoted as KD), PKD, and TKD. 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 $\{0.2,\ 0.5,\ 0.7, 0.9\}$, and the TKD weight γ as $\{10,\ 50,\ 100,\ 500,\ 1000\}$. 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.

5.3 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-turn. "BERT_{12} (Teacher)" denotes our fine-tuned teacher model. "BERT_3" de-

[1] https://huggingface.co

notes our fine-tuned student model without using knowledge distillation. "$BERT_3$-X" denotes the trained student model via the knowledge distillation approach "X", i.e., vanilla knowledge distillation (KD), PKD, TKD, and the fusion of both PKD and TKD (Fusion).

In Table 1, besides the fusion approach, $BERT_3$-TKD achieves competitive results in all datasets, and $BERT_3$-Fusion outperforms all other approaches. We also find out that there is little increase in the computational cost of using TKD than PKD. The experiment results show that TKD achieves competitive performance for compressing BERT, among other knowledge distillation methods in GLUE. TKD can be fruitfully combined with PKD to achieve better performance.

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 2.4% and 2.7% of the accuracy, and improves $BERT_3$-KD by 0.5% and 0.7%. For SST-2, $BERT_3$-TKD improves $BERT_3$ by 1.7%, improves $BERT_3$-KD by 1.2%, and improves $BERT_3$-PKD by 0.6%. For RTE, $BERT_3$-TKD improves $BERT_3$ by 3.2%, improves $BERT_3$-KD by 2.2%, and improves $BERT_3$-PKD by 0.2%. For QQP, $BERT_3$-TKD improves $BERT_3$ by 2.0%, improves $BERT_3$-KD by 0.3%, and improves $BERT_3$-PKD by 0.1%. On MPRC, we found that the fine-tuned student model has already achieved a good performance, then there is little progress it can obtain from the distillation of the teacher.

5.4 Fusion Approach

The knowledge distillation community has made several independent improvements for compressing language models. However, it is unclear which of these extensions are complementary and can be fruitfully combined. Here we examine the combination of PKD and TKD. Note that, KD is already considered in either TKD or PKD. Then the combination can be considered as the fusion of KD, PKD, and TKD. In specific, the overall loss function for the fusion approach is defined as:

$$L_{Fusion} = (1-\alpha)\, L_{DS} + \alpha\, L_{CE} + \beta\, L_{PD} + \gamma\, L_{TD}. \quad (5)$$

The experimental results are shown in Table 1, which show that TKD can be fruitfully combined with PKD to achieve better performance. We found that when the size of the dataset is larger, better improvement of the performance would be observed. In specific, compared with $BERT_3$-TKD, $BERT_3$-Fusion improves the performance by 1.6% on SST-2, 1.8% on MNLI-m, 0.7% on MNLI-mm, and 0.2% on QQP.

5.5 Discussion

Experimental results in Table 1 show that TKD and the fusion approach provide better improvements for the performance of the fine-turned student model on datasets with the larger size. This observation implies that TKD and the fusion 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. Experimental results show that this 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 Wang et al. [2020]. In the knowledge distillation procession, the student intents 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 to some subgraphs in the knowledge graph for the teacher. Intuitively, it is more helpful if such subgraphs 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 encourages 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 subgraphs 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 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.

Model	SST-2 (67k)	MPRC (3.7k)	MNLI-m (393k)	MNLI-mm (393k)	RTE (2.5k)	QQP (364k)
$BERT_{12}$ (Google)	93.5	88.9/84.8	84.6	83.4	66.4	71.2/89.2
$BERT_{12}$ (Teacher)	94.3	89.2/85.2	83.7	82.8	69.1	70.9/89.0
$BERT_3$	86.4	80.5/72.6	74.8	74.3	55.2	65.8/86.9
$BERT_3$-KD	86.9	79.5/71.1	75.4	74.8	56.2	67.3/87.6
$BERT_3$-PKD	87.5	80.7/72.5	76.7	76.3	58.2	68.1/87.8
$BERT_3$-TKD	88.1	80.7/72.5	77.2	77.0	58.4	68.1/87.9
$BERT_3$-Fusion	**90.7**	**80.7/72.6**	**79.0**	**77.7**	**58.5**	**68.1/88.1**

Table 1: Different student models on GLUE benchmark. The best results for student models are in-bold. Google's submission results are obtained from official GLUE leaderboard. $BERT_12$ (Teacher) is our own implementation of the teacher model. PKD's results are obtained from its paper.

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 own 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 GLUE benchmark. Moreover, experimental results also show that TKD can be fruitfully combined with other existing knowledge distillation 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

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Zehao Huang and Naiyan Wang. 2017. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Seunghyun Lee and Byung Cheol Song. 2019. Graph-based knowledge distillation by multi-head attention network. *arXiv preprint arXiv:1907.02226*.

Junjie Liu, Dongchao Wen, Hongxing Gao, Wei Tao, Tse-Wei Chen, Kinya Osa, and Masami Kato. 2019a. 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *Proceedings of the 3rd International Conference on Learning Representations, (ICLR-2015)*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. 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.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. 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.

Frederick Tung and Greg Mori. 2019. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1374.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. 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.

Sergey Zagoruyko and Nikos Komodakis. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*.