AMP:the Attention Mechanism of Multiple Prompts for Transfer Learning

Anonymous ACL submission

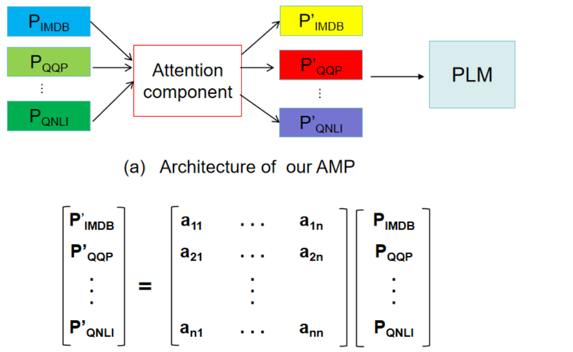
Abstract

Prompt transfer learning can significantly improve the performance of prompt-tuning methods. However, it requires much manual work to find out the proper source tasks which can yield positive transfer for the target task. We propose a two-stage multiple prompts transfer learning approach called AMP to address this drawback. First, we train a source prompt for each task as task embedding. Second, we learn a target prompt for each task which is an attention-weighted sum of source prompts through training an attention component. The attentions control the influence each source task yields for the target task, through which proper source tasks for the target task can be automatically identified. A source prompt is a 2D matrix, but the traditional attention mechanism only receives vectors. The prior methods employ pooling or flattened method to transform the matrix to the vector for computing the attentions between a set of matrices. We propose a method called DAM which can compute attentions between matrices directly without transforming. DAM method can more exactly compute the attentions between matrices. Wide experiments demonstrate that AMP is effective and can improve the performance of prompt-tuning without any prior search.

1 Introduction

In earlier years, the most commonly used approach is to fine-tune the entire pretrained language models(PLMs) for NLP tasks(Devlin et al. (2018);Liu et al. (2019);Lewis et al. (2019);Yang et al. (2019);Bao et al. (2020)). Although fine-tuning method achieves state-of-the-art performance, it requires to update all parameters of PLMs and store a large specific-task model for each task.

Recently, many studies focus on prompt-tuning method which learns a small number of prompt tokens for each task on frozen PLMs(Liu et al. (2021b);Chen et al. (2022);Qin and Eisner (2021);Han et al. (2022)). It only updates the prompt parameters but keeps PLMs fixed during training. It merely stores a specified small prompt for each task and the backbone PLMs are shared across all tasks. However, prompt-tuning methods decrease task performance and are sensitive to prompt initialization(Lester et al. (2021);Liu et al. (2021a);Gu et al. (2021)).



(a) Architecture of our AMP

(b) Each target prompt is an attention-weighted sum of source prompts

Figure 1: An illustration of our AMP method. (a): We combine source prompts to learn an attention component to obtain a target prompt for each task. (b): Each target prompt is an attention-weighted sum of source prompts. The learned attentions control the influence each source task yields for the target task.

Some literatures(Vu et al. (2021);Gu et al. (2021);Asai et al. (2022)) propose prompt transfer learning to solve these shortcomings. When it starts to learn a target task, it firstly learn a source prompt on one or more source tasks similar with the target task and then use the source prompt to initialize the target prompt. It transfers the knowledge of source tasks to the target task and improves the performance of the target task. However, it requires extensive test or considerable manual computation to find out source tasks which can yield positive transfer for a target task.

In this paper,we propose a two-stage multiple prompts transfer learning approach called AMP which is illustrated in Figure 1. In first stage, AMP trains a source prompt for each task as task embedding on a frozen PLM. In second stage, AMP learns an attention component to compute the attentions between source prompts. Given the attentions, a new prompt for each task is calculated as attention-weighted sum of source prompts. We call this prompt as target prompt. The attentions control influence each source task yields for the target task. A high attention is learned if a source task can yield positive influence for the target task. Otherwise, a low attention is learned. This can make AMP to automatically identify source tasks which yield positive transfer for the target task.

The attention mechanism is always exploited on an input matrix consisted of a set of vectors(Vaswani et al. (2017);Devlin et al. (2018);Liu et al. (2019);Radford et al. (2018)). It firstly projects the input matrix into three matrices–queries,keys and values and then calculates the attentions between each query vector and all key vectors through the dot product. Finally, an output matrix is obtained where each vector is attention-weighted sum of value vectors. However, this procedure is unable to compute attentions between a set of matrices directly. The prior methods transform the matrix into the vector before computing attentions. The widely used methods are the pooling method which computes average or maximum of each dimension to obatin the vector and flattened method which reshapes a matrix into a sequence(Asai et al. (2022);Dosovitskiy et al. (2020);Wang et al. (2021);Chu et al. (2021)). The pooling method causes some details lost and the flattened method destroys the original structure. They can't express attentions between matrices exactly.

A source prompt is a 2D matrix. We introduce a new method called DAM to compute the attentions between source prompts. It can more exactly compute attentions between a set of matrices.

We empirically evaluate our AMP method on diverse tasks. The experimental results show that AMP can automatically finds out right source tasks for target task and largely improves the performance of prompt-tuning method.

2 Background

In this section, we give a brief overview of common methods in NLP. This is followed by our goal.

Task definition. We define a set of n tasks: $C = \{T_1, T_2, ..., T_n\}$. The aim is to share knowledge between tasks to improve the performance of each task with low training and storing cost.

Transfer learning. A masked language model is pretrained on large corpus of unlabelled text(called PLM). When learning a specified task, PLM is transferred to the task and the full parameters of PLM are fine-tuned on the task(Devlin et al. (2018);Liu et al. (2019);Yang et al. (2019);Lan et al. (2019);He et al. (2020)). An independent model is obtained for each task.

$$\begin{aligned} \theta_1 &\leftarrow \underset{\theta}{argmin}\, L(T_1; \theta) \\ &\vdots \\ \theta_n &\leftarrow \underset{\theta}{argmin}\, L(T_n; \theta) \end{aligned}$$

where, L denotes as the loss function and θ represents the parameters of PLM.

It aims to make each task benefit from the knowledge stored in PLM.

Multi-task learning. All tasks are trained simultaneously on a PLM((Liu et al. (2016);Liu et al. (2017); Ruder (2017);Sanh et al. (2019);Zhang and Yang (2021))). A shared model is learned for all tasks.

$$\theta^{'} \leftarrow \underset{\theta}{argmin} \sum_{i=1}^{n} L(T_i; \theta)$$

Only a shared mode is stored for all tasks. More importantly, it can make all tasks benefit from each other. However, all tasks have to be prepared well before training. If a new task is added after training, it will have to access all tasks to retrain the model from scratch.

Prompt-tuning. It adds some prompt tokens into the task. Then the task is fed into a PLM to train. Only those prompt parameters are updated during training, but the PLM is kept fixed. It learns a separated prompt for each task, but PLM is shared across all tasks((Li and Liang (2021);Liu et al. (2021b);Qin and Eisner (2021)).

$$\begin{aligned} \phi_1 &\leftarrow \underset{\phi}{argmin}\, L(T_1; \phi, \theta) \\ &\vdots \\ \phi_n &\leftarrow \underset{\phi}{argmin}\, L(T_n; \phi, \theta) \end{aligned}$$

where, ϕ denotes the prompt parameters. ϕ is much smaller than θ. So, prompt-tuning does with low training and storing cost. However, it always performs not better than full-parameters tuning and is sensitive to prompt initialization(Lester et al. (2021);Gu et al. (2021)).

Prompt transfer learning. When to learn a prompt for a target task T_i , it firstly learns a source prompt $\phi^{'}$ on one or more tasks and then uses $\phi^{'}$ to initialize the target prompt(Vu et al. (2021);Gu et al. (2021);Sanh et al. (2021);Min et al. (2021)).

$$
\begin{aligned}
T_s &= \bigcup_{1 \leq j \leq m, j \neq i} T_j, m \in \{1, \ldots, n\} \\
\phi^{'} &\leftarrow \underset{\phi}{argmin}\, L(T_s, \theta) \\
\phi_i &\leftarrow \underset{\phi}{argmin}\, L(T_i; \phi^{'}, \theta)
\end{aligned}
$$

Vu et al. (2021) shows that when target prompt is initialized by right source prompt, the performance of prompt-tuning methods can be largely improved.

It is, however, difficult to find out the right source tasks for a target task. Because the relationships between tasks are extremely complexed. Intuitively, source tasks which are same type as target task can yield positive transfer for target task. But Vu et al. (2021) suggests that some source tasks which are different type with target task can also yield positive transfer. More seriously, even through some source tasks are same type with target task, they yield negative transfer. It requires to test source task one by one to find out a set of right source tasks for the target task.

Vu et al. (2021) proposes a method which interprets learned prompt for each task as task embedding and similarity between tasks is defined as the cosine similarity score between task prompts. Vu et al. (2021) shows that the source tasks which have high similarity scores with the target task can yield positive transfer in general. It doesn't require massive test, but it requires much manual work to compute the similarity scores between target task and each source task . Additionally, negative transfer still occurs between tasks with high similarity scores.

ATTEMPT(Asai et al. (2022)) can automatically find out proper source prompts for each example in target task through computing the attention between the example embedding and source prompts. However, it can't express the relationships between the whole target task and source tasks. Additionally, it has to retrain all tasks when a new task is added after training.

Our goal. We hope to not only achieve to transfer knowledge from source tasks to target task, but also how much influence each source task yields for target task is exactly expressed. We hope to automatically identify right source tasks which yield positive transfer for target task. We also hope to flexibly add a new task.

3 Method

In this section, we show our AMP method in detail. AMP trains a source prompt for each task in the first stage(§3.1). Then it combines all source prompts to train an attention component, through which a target prompt for each task is learned(§3.2). We propose DAM method to compute the attention of matrices during learning target prompt(§3.2.1). Subsequently, we give an efficient implementation method of DAM detailedly(§3.2.2). Finally, we show inference process of AMP(§3.3) and how to add a new task(§3.4).

3.1 Source Prompt

In first stage, we train a source prompt for each of n tasks on a frozen PLM as the task embedding. The length of all source prompts is set to be same. We obtain n source prompts $\{P_1, ..., P_n\}$,where $P_i \in \mathbb{R}^{l*d}$, l is prompt length and d is model dimension of PLM. n prompts are packaged into a 3D matrix $P \in \mathbb{R}^{n*l*d}$.

3.2 Target Prompt

In second stage, we put an attention component ψ on top of PLM. The attention component takes P as the input. We calculate the attentions between source prompts through ψ. Then we obtain n target prompts $\{P_1^{'}, ..., P_n^{'}\}$, each of which is an attention-weighted sum of source prompts as followed.

$$
\begin{bmatrix} P_1^{'} \\ \vdots \\ P_n^{'} \end{bmatrix} = \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ & \vdots & \\ a_{n1} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix}
$$

Each task is prefixed with a correspond target prompt and then the task is fed into the PLM to train again. During training, only ψ is updated, while the source prompts P and PLM are kept fixed. ψ is trained by all tasks simultaneously.

The attentions represent the influence each source task yields for target task. A high attention is learned if a source task can yield positive influence for the target task. Otherwise, a low attention is learned.

3.2.1 Attention Component

The attention component ψ consists of three projection parameter matrices $W^Q \in \mathbb{R}^{d*k}$, $W^K \in \mathbb{R}^{d*k}$ and $W^V \in \mathbb{R}^{d*v}$,where d is the model dimension of PLM, k is the dimension of queries and keys, v is the dimension of values and v is equal to d . The input P is projected into three 3D matrices– queries $Q \in \mathbb{R}^{n*l*k}$, keys $K \in \mathbb{R}^{n*l*k}$ and values $V \in \mathbb{R}^{n*l*v}$, where each query and key are a 2D matrix.

We propose DAM method to calculate the attention of a query-key pair (q, k).

$$atten(q,k) = \frac{1}{l^2}\sum_{i}^{l}\sum_{j}^{l}(a_i \bigotimes b_j)$$

where, $\bigotimes$ represents dot product, a_i and b_j denote a vector in q and k, respectively. It calculates the dot product between each vector in query q and that in key k, so it is more exact. It is illustrated in Figure 7.

3.2.2 Implementation Details of DAM

DAM is implemented in following 4 steps.

Firstly, we reshape $P \in \mathbb{R}^{n*l*d}$ into matrix $P^{'} \in \mathbb{R}^{m*d}$, where $m = n*l$. $P^{'}$ is linearly projected to obtain the matrix Q, K and V .

$$\begin{aligned} P^{'}W^Q &= Q \in \mathbb{R}^{m*k} \\ P^{'}W^K &= K \in \mathbb{R}^{m*k} \\ P^{'}W^V &= V \in \mathbb{R}^{m*v} \end{aligned}$$

Secondly, We calculate the attentions between queries and keys.

$$QK^T = S \in \mathbb{R}^{m*m}$$

S is divide into $n*n$ blocks,where the size of each block is $l*l$.

$$S = \begin{array}{c} \\ q_1 \\ \vdots \\ q_n \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & \cdots & k_n \end{array} \\ \left(\begin{array}{c:c:c} b_{11} & \cdots & b_{1n} \\ \hdashline \vdots & \ddots & \vdots \\ \hdashline b_{n1} & \cdots & b_{nn} \end{array}\right) \end{array}$$

the block b_{ij} represents dot products between each vector in query q_i and that in key k_j.

$$b_{ij} = \begin{array}{c} \\ q_i^1 \\ \vdots \\ q_i^l \end{array} \begin{array}{c} \begin{array}{ccc} k_j^1 & \cdots & k_j^l \end{array} \\ \begin{pmatrix} a_{11} & \cdots & a_{1l} \\ \vdots & \ddots & \vdots \\ a_{l1} & \cdots & a_{ll} \end{pmatrix} \end{array}$$

Thirdly, we leverage convolution operator on S to get the sum of each block. The size of convolution kernel is set to $l*l$, which is same as that of block. The stride size is set to l. The kernel value is set to1. We obtain a matrix $S^{'} \in \mathbb{R}^{n*n}$. Then $S^{'}$ is scaled by $1/l^2$. A softmax function is leveraged on $S^{'}$.

$$S^{'} = \begin{array}{c} \\ q_1 \\ \vdots \\ q_n \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & \cdots & k_n \end{array} \\ \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \end{array}$$

where $S^{'}_{ij}$ is the attention between query q_i and key k_j.

Fourthly, $V \in \mathbb{R}^{m*v}$ is reshaped into $V^{'} \in \mathbb{R}^{n*l*v}$. Then we multiply $V^{'}$ by $S^{'}$ to obatin the output matrix $O \in \mathbb{R}^{n*l*v}$, where n target prompts is earned and the length and dimension of target prompt are l and v,respectively. Each target prompt corresponds a task.

3.3 Inference

After training, we obtain a target prompt for each task . The source prompts and the attention component are no longer needed. The target prompt is concatenated to the input embedding to form the input sequence. Then the input sequence is fed into PLM to acquire the final result. The inference process is same as in prompt-tuning. AMP doesn't increase extra inference cost.

3.4 Adding a new task

When a new task is added after training original tasks, AMP firstly learns a source prompt for the new task and then combines all source prompts to train the attention component to obtain a target prompt for the new task. As the attention component is not used during inference, the inference process of original tasks isn't affected. AMP doesn't require complete re-training when a new task is added.

4 Experiments

We conduct experiments on 11 NLP tasks across diverse types to evaluate the performance of our AMP method in this section. Those tasks and the related datasets are shown in §4.1. The experimental setup is described in §4.2.

4.1 Tasks

We briefly list the tasks used in our experiment. A detailed description about those tasks is shown in Appendix §A.4.

Sentiment analysis predicts whether a sentence to be positive or negative: IMDB(Maas et al. (2011)), SST-2(Socher et al. (2013)), Yelp-2(Zhang et al. (2015)). **Sentence relatedness** predicts whether one sentence is similar with the other or not: STS-B(Cer et al. (2017), MRPC(Dolan and Brockett (2005)). **Entailment** predicts whether two sentences entail or contradict: RTE(Giampiccolo et al. (2007)), SciTail(Khot et al. (2018)), CB(De Marneffe et al. (2019)). **Question answering** predicts the right answers for some questions after reading a passage: MultiRC(Khashabi et al. (2018)), BoolQ(Clark et al. (2019)), QNLI(Wang et al. (2018)).

4.2 Experimental Setup

Source prompt training. We use RoBERTa-base(Liu et al. (2019)) as PLM. We adopt the AdamW optimizer. The learning rate is set 10^{-4} with a linear decay. We set the maximum training epochs to 30 with early stopping. The length of prompt tokens is set 100 for all tasks. Each prompt is initialized by randomly sampling tokens from common vocabularies.

Attention component training. We still use RoBERTa-base as PLM. The maximum training epochs is set to 10. The learning rate is set $5 * 10^{-5}$ with a linear decay. The maximum token length is set to 384 for all tasks. We combine the datasets of all tasks together to train the attention component using examples-proportional strategy(Raffel et al. (2020)), where the maximum training examples are limited to 100K for each task.

In attention component, v is set to 768 which is the model dimension of RoBERTa-base and k is set to 768.

Baselines. We compare AMP with fine-tuning, prompt-tuning, SPoT and ATTEMPT. SPoT adopts two strategies: SPoT-s and SPoT-m. SPoT-s initializes target prompt with similarity-weight average of all source prompts. SPoT-m initializes target prompt with a source prompt learned on MNLI task which is proven to be able to improve the performance for most target tasks (Vu et al. (2021)).

Max-pooling method for computing attentions. We make a comparison between DAM method and max-pooling method for computing attentions between source prompts. The max-pooling method takes the same steps as DAM except the computation of attentions. It firstly obtains the query matrices, key matrices and value matrices as the first step of DAM. Then each query and key is translated into a vector through performing max-pool operation for each dimension. The dot product between each query and key is calculated to obtain the attention matrix. The following steps are same as the DAM method.

5 Result

We show the main result in §5.1. We present the the effectiveness of DAM in §5.2.

5.1 AMP

As illustrated in Table 1, AMP outperforms prompt-tuning, SPoT-s and ATTEMPT. There are five findings as followed.

(1) AMP outperforms prompt-tuning by a large margin. AMP improves performance for 9 out of 11 tasks. This shows that AMP can find out right source tasks for most target tasks.

(2) AMP performs better than SPoT-s. AMP doesn't achieve improvement of performance for 2 out of 11 tasks, but SPoT-s doesn't increase performance for 5 tasks. This shows that the attentions learned dynamically are more reliable than constant similarity scores for finding right source tasks.

(3) AMP performs lower than SPoT-m. AMP doesn't conduct prior massive search which is required to SPoT-m. AMP outperforms ATTEMPT.

(4)We observe that AMP is more beneficial for small datasets than large datasets. AMP achieves improvement of 6.3% for MultiRC(5.1k) and 6.6% for BoolQ(9.4k) , but it only increases 1.8% and 3.1% performance for IMDB(25k) and SST-2(67k) respectively . This shows that AMP can find more right source tasks for small task.

(5)We also find that AMP can match fine-tuning for 2 tasks. AMP helps close the gaps between prompt-tuning and fine-tuning.This indicates that

Dataset	fine-tuning	prompt-tuning	SPoT-s	SPoT-m	ATTEMPT	AMP
IMDB	93.1	86.5	85.2	91.8	90.3	88.3
SST-2	90.1	86.8	87.5	87.1	86.3	89.9
Yelp-2	88.4	83.5	81.2	84.9	83.1	83.9
STS-B	86.5	81.2	83.5	85.2	83.4	86.3
MRPC	87.9	69.4	68.5	74.1	73.3	76.6
RTE	71.1	57.8	66.4	68.8	68.1	66.8
SciTail	93.3	87.8	86.2	88.1	86.3	86.9
CB	83.5	71.4	75.3	84.1	81.3	78.9
MultiRC	73.1	64.4	74.1	76.2	70.1	70.7
BoolQ	75.8	63.5	69.4	72.2	68.2	70.1
QNLI	89.5	85.4	84.3	86.2	85.1	83.3
Mean	84.8	76.2	78.3	81.7	79.6	80.2

Table 1: Results of different tuning methods. All results are based on RoBERTa-base. The results are Pearson Correlation for STS-B , F1 score for MultiRC and accuracy score for others. The ATTEMPT represents shared ATTEMPT.

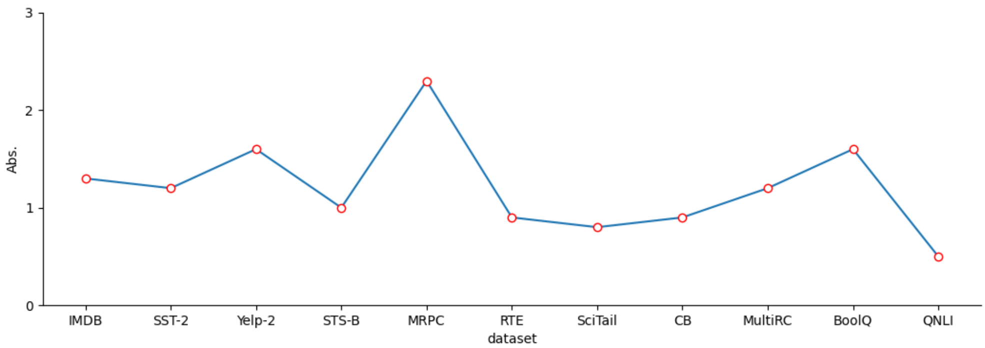


Figure 2: Absolute imporvement of DAM over max-pooling.

prompt-tuning method has potential to outperform fine-tuning method through transferring right source tasks to target task .

5.2 DAM

Figure 2 shows the improvement of performance of DAM over max-pooling method. We can find that DAM method exceeds max-pooling method by a margin. At the best, DAM improves 2.3% performance. This shows that DAM could be more helpful for improving performance of task.

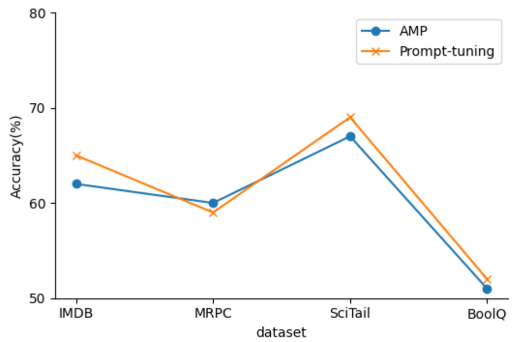


Figure 3: Performance of AMP on RoBERTa-small model

6 Analyses

Scale of PLM. The size of parameter matrix W^Q,W^K and W^V is controlled by model dimension of PLM. So, we think that AMP is largely affected by PLM. We evaluate AMP on small PLM. As illustrated in Figure 3, AMP perform worse than prompt-tuning .

Dimension of queries and keys. We evaluate the performance of AMP with different k $\{512, 256, 64\}$. As illustrated in Figure 4, the performance of AMP decreases as k becomes small. This indicates that it is important to project source prompt into high-dimensional space for perfor-

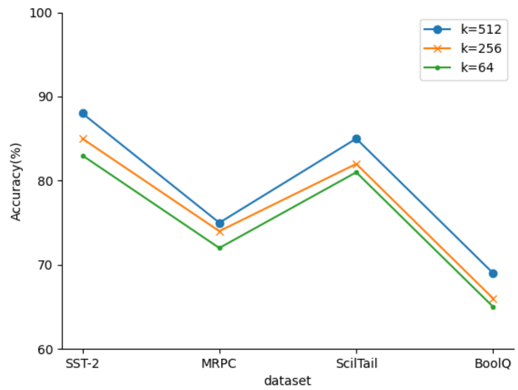


Figure 4: Performance of AMP on different dimension of queries and keys

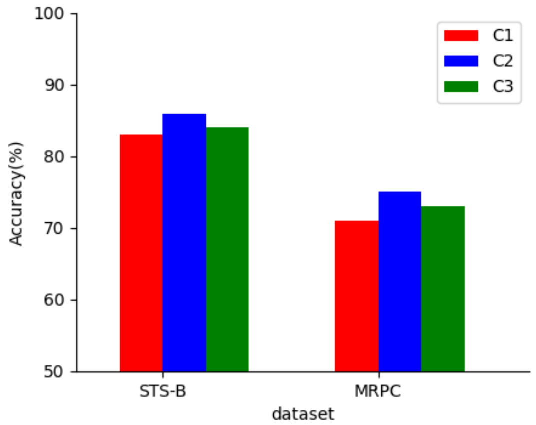


Figure 5: Performance of AMP under different task sets. C1:{STS-B, MPRC}, C2:{STS-B, MPRC, QNLI},C3:{STS-B, MRPC, SST-2}

mance of task.

Different task sets We empirically analyze how different task sets affect the performance of AMP. The result is shown in Figure 5. We find that the performance of the same task change with task sets. The right source tasks for a target task are not same in different task sets. This shows that proper source tasks play an important role for the performance of target task.

Attention visualization. Figure 6 is the attention matrix learned by AMP. In general, AMP gives a high attention for two same type of tasks, for example IMDB and Yelp2, STS-B and MRPC, RTE and SciTail.

The task MRPC highly attend QNLI, but they are different type . Similar phenomenon also appears between RTE and QNLI ,STS-B and QNLI. In-

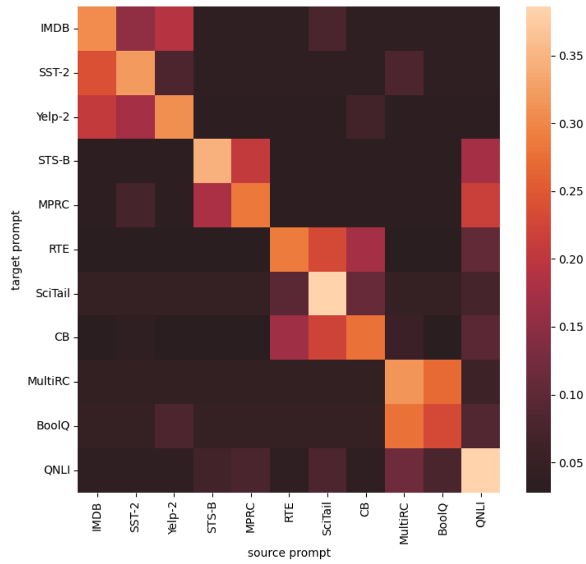


Figure 6: Attentions between target tasks(row) and source tasks(column).

versely, even though QNLI is same type with MulitiRC , but QNLI is lowly attended by MulitiRC . This shows that AMP can find out the implicit relationships between tasks.

7 Related Work

Parameter-efficient transfer method .Adapter(Houlsby et al. (2019);Karimi Mahabadi et al. (2021);Rücklé et al. (2020);Hu et al. (2021)) inserts a small learnable module into the PLM. It only trains the module while keeps PLM fixed during training.BitFit(Zaken et al. (2021)) only updates the biases of PLM for each task.Pfeiffer et al. (2020) proposes AdatperFusion to improve the performance of Adapter and achieve the multi-task learning.

Recently, learnable soft-prompt methods(Liu et al. (2021b);Li and Liang (2021);Lester et al. (2021);Zhang et al. (2021)) have gradually replaced early hard-prompt methods(Schick and Schütze (2020);Gao et al. (2020);Shin et al. (2020);Jiang et al. (2020)).

In concurrent work, (Vu et al. (2021);Gu et al. (2021);Asai et al. (2022)) also explore prompt transfer methods. Gu et al. (2021) pretrain a prompt on 10GB data and then transfer the prompt to target task. Vu et al. (2021) requries much computation to find the right source tasks for a target task. However, our work mainly focuses on automatically searching right source tasks for a target task.

Multi-task transfer learning methods. Recent approaches train a large model on massive tasks.

Then the model is transferred to unseen tasks without updating any parameter (Talmor and Berant (2019);Sanh et al. (2021);Wang et al. (2022);Mishra et al. (2021);Wei et al. (2021);Gupta et al. (2022);He et al. (2021);).They focus on traning a unified model which can be applied in any NLP task.

8 Conclusion

We present a multi-prompt transfer learning approach called AMP. AMP exactly computes the influence each source task yields for the target task and can automatically identify right source tasks for the target task. AMP largely improves performance of promp-tuning, while it doesn't increase extra inference cost. AMP can flexibly add new task without complete retraining. Additionally, We propose a DAM method which can exactly compute the attentions between a set of matrices. Finally, we visual the attention matrix to show that AMP can reveal the implicit relationships between tasks.

Limitations

Our method has three main limitations. First, AMP has to train twice for each task. This increases training time. It combines multiple tasks to train the attention component, which increases the training difficulties. Secondly, it requires that the maximum tokens for each task must be same in the second stage. It has to make trade-off between memory and performance. Thirdly, the computation cost of DAM increases exponential times compared to max-pooling method. DAM method is not suitable for computing the attentions between large matrices.

References

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International conference on machine learning*, pages 642–652. PMLR.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. 2021. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed H Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689*.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. 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 252–262.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

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

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2020. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

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.

Alon Talmor and Jonathan Berant. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. *arXiv preprint arXiv:1905.13453*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv e-prints*, pages arXiv–2204.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Differentiable prompt makes pre-trained language models better few-shot learners. *arXiv preprint arXiv:2108.13161*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

A Appendix

A.1 Number of parameters of AMP

source prompt training. The source prompts are trained through general promtp-tuning method. The number of parameters to be updated are $n * l$, where n is prompt length and l is dimension of PLM.

Attention component training. The attention component ψ consists of three parameter matrices $W^Q \in \mathbb{R}^{d*k}$,$W^K \in \mathbb{R}^{d*k}$ and $W^V \in \mathbb{R}^{d*v}$, where v is equal to d. The number of parameters is $2dk + d^2$.

Number of parameters of different tuning methods As illustrated in Table 2, the number of parameters of AMP is less than 1.4% of those of Fine-tuning.

method	parameters
Fine-tuning	125M
prompt-tuning	77k
SPoT	77k
ATTEMPT	232K
AMP	1.8M

Table 2: Number of parameters of different method based on RoBERTa-base model. The prompt length is set 100.

A.2 Method Details

DAM vs max-pooling We visual DAM and max-pooling method in Figure 7. We can see that why DAM is more exactly calculate the attentions than max-pooling.

Inference process After traing, according to section 3.2.2, a group of target prompts are obtained, each of which corresponds to a task. The target prompt is used to inference instead of source prompt. Source prompts and the attentiom componen are not used during inference.This brings two benefits: first, it doesn't increase any inference time than prompt-tuning method and only increase prompt computation than fine-tuning method;second, when a new task is added after training, the attention component must be updated. Howerver, the inference process of the original task is not affected.So, AMP can flexibly add a new task.

A.3 Hyperparameters

We conduct search on the hyperparameters including learning rate $\{10^{-4}, 5*10^{-4}, 10^{-5}, 5*10^{-5}\}$, training epoches $\{10, 30, 50\}$, batch size $\{4, 8\}$. The search doesn't be conducted on all tasks. We choose SciITail and BoolQ to obtain the best hyperparameter setup. All experiments are performed on a single GPU. The results are reported on validation sets except IMDB, Yelp-2 and SciITail. Those three tasks are reported on test sets. For each task, we run for 3 times and the best result is reported.

AMP The maximum token length for source prompt traning can be different. It is set to 348 for MultiRC, 256 for other task. The maximum token length for target prompt traning must keep same for all tasks. We set it to be 384. We set weight decay to be 10^{-5}. The warm step is set to 500.

SPoT. SPoT-m: (Vu et al. (2021) shows that the task MNLI has good transferability and can improve the performance for most task. A source task is firstly obtained on MNLI task and then is used to initialize the target prompt for each task in our task sets. SPoT-s: We obtain a source prompt for each task as prompt-tuning. The similarities between tasks are obtained through the average cosine similarity between prompt token in (Vu et al. (2021). The other settings is same as those in AMP.

ATTEMPT We don't use the prompt for large-scale datasets as source prompt like Asai et al. (2022). We train a source prompt for each task in our task sets and then transfer them to other tasks. This is in line with AMP.

A.4 Task sets

To verify whether AMP can automatically identify right source tasks for target task, we sample 11 task from a collection of NLP tasks without any prior bias choice. Among those tasks, 4 tasks are from GLUE and the other 4 tasks are from SuperGLUE. We list those tasks detailedly in Table 3.

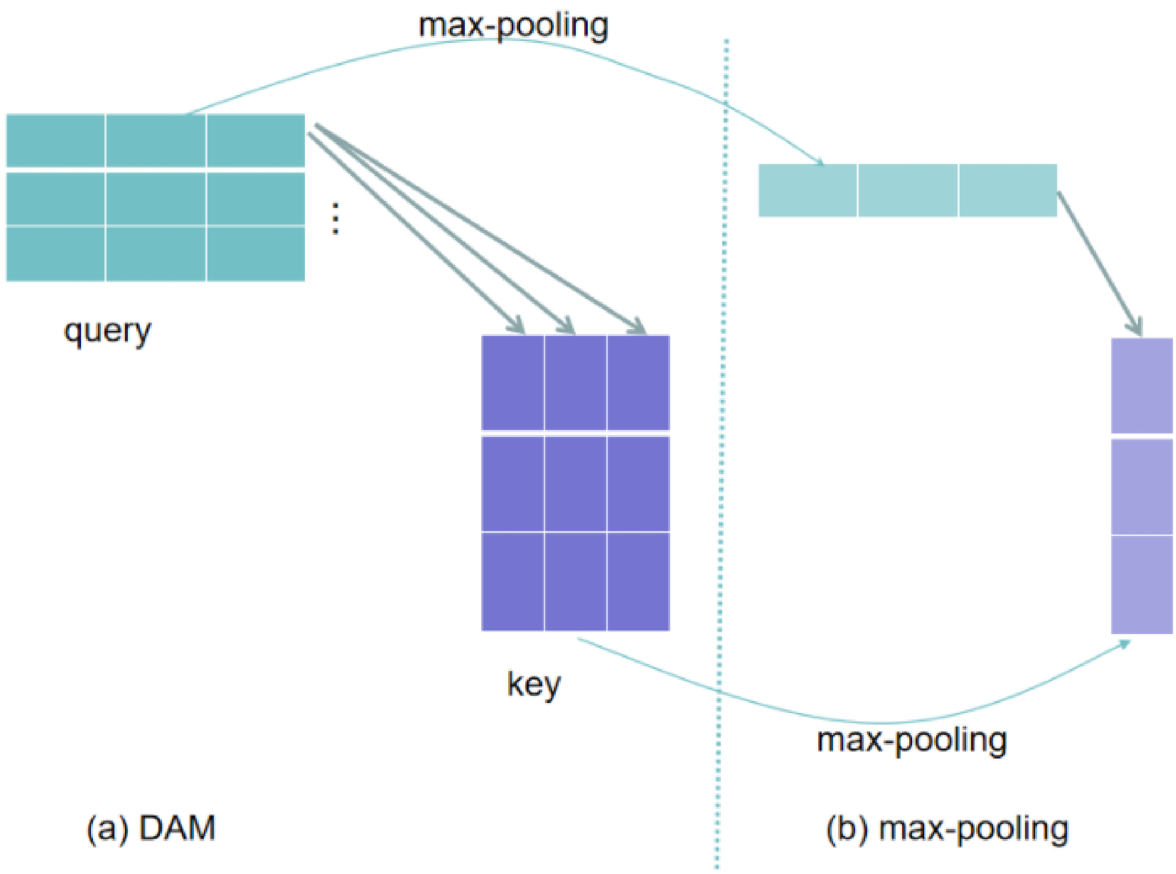


Figure 7: DAM vs max-pooling for compting the attention between matrices.

Task	soure	type	metric	input type	result type
IMDB	others	Sentiment analysis	accuracy	single sententce	positive/negative
SST-2	GLUE	Sentiment analysis	accuracy	single sentence	positive/negative
Yelp-2	others	Sentiment analysis	accuracy	single sentence	positive/negative
STS-B	GLUE	Sentence relatedness	Pearson corr.	sentence-pair	similarity score(1-5)
MRPC	GLUE	Sentence relatedness	accuracy	sentence-pair	equivalent/not equivalent
RTE	GlUE	Entailment	accuracy	text-hypothesis	entailment/not entailment
SciTail	others	Entailment	accuracy	text-hypothesis	entailment/not entailment
CB	SuperGLUE	Entailment	accuracy	text-hypothesis	entailment/not entailment
MultiRC	SuperGLUE	Question answering	F1	a paragraph a question a list of answers	each answer is ture or false
BoolQ	SuperGLUE	Question answering	accuracy	question-paragraph pair	yes or no
QNLI	GLUE	Question answering	accuracy	question-paragraph pair	answer is contained in paragraph or not

Table 3: The details of 11 tasks. The metrics are those used in our experiments.