AdapterDistillation: Non-Destructive Task Composition with Knowledge Distillation

Junjie Wang[*], **Yicheng Chen**[*], **Wangshu Zhang, Sen Hu, Teng Xu, Jing Zheng**
Ant Group
{benge.wjj, yicheng.chen, wangshu.zws, hs272483, harvey.xt, jing.zheng}@antgroup.com

Abstract

Leveraging knowledge from multiple tasks through introducing a small number of task specific parameters into each transformer layer, also known as adapters, receives much attention recently. However, adding an extra fusion layer to implement knowledge composition not only increases the inference time but also is non-scalable for some applications. To avoid these issues, we propose a two-stage knowledge distillation algorithm called AdapterDistillation. In the first stage, we extract task specific knowledge by using local data to train a student adapter. In the second stage, we distill the knowledge from the existing teacher adapters into the student adapter to help its inference. Extensive experiments on frequently asked question retrieval in task-oriented dialog systems validate the efficiency of AdapterDistillation. We show that AdapterDistillation outperforms existing algorithms in terms of accuracy, resource consumption and inference time.

1 Introduction

Recently task-oriented dialogue systems have found extensive applications in diverse business domains (Yan et al., 2017; Wei et al., 2018; Valizadeh and Parde, 2022). Owing to the idiosyncratic features of these domains, custom dialogue systems are often required. Nonetheless, the fundamental functions and architectures underlying these systems typically exhibit noteworthy similarities. Hence, the adoption of a platform-based strategy for accommodating task-oriented dialogue systems across multiple domains has emerged as a promising and effective approach.

One popular method is called Multi-Task Learning (MTL), which aims to train multiple tasks simultaneously based on the shared representation of all tasks as shown in Figure 1, resulting in relatively good performance on each task (Collobert and Weston, 2008; Chen et al., 2022b,a). However, new tenants often register on the platform in a streaming manner. Therefore, predictions for the existing tenants in MTL would be compromised when new tenants are added to the platform since retraining often occurs at that time.

*Equal Contributions

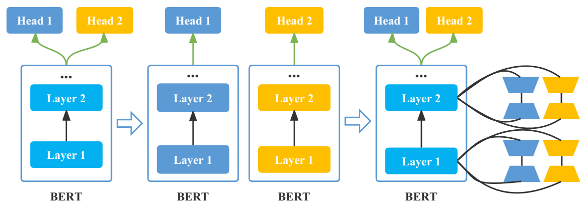

Figure 1: Three popular multi-tenant learning methods. Left: Multi-Task Learning. Middle: Independent Learning. Right: Adapter Learning.

To ensure that tenants do not interfere with each other, an intuitive approach is to train a task-specific model for each tenant. However, this independent training approach requires a significant amount of resources to store complete model parameters. It is clear that the resource consumption becomes the bottleneck as many tenants register on the platform. Additionally, fine-tuning all parameters on a tenant with very little data can often lead to severe overfitting (Dietterich, 1995; Hawkins, 2004). Thus, providing tenants with the ability to solve designated tasks with limited resources is necessary.

Owing to the distinctive properties of platform-based systems, tasks implemented on the platform had better satisfy the following criteria: 1) The platform witnesses a continuous influx of new tenants. It is incumbent upon the model to ensure the performance of the existing tenants are not destructed when new tenants are added. 2) The resources of the platform are limited, thereby necessitating the provision of tenants with the capability to ensure the task performance of each tenant with minimal storage and computational resources. Given this practical limitation, for an incoming tenant, how to utilize the current tenant data and the existing tenant models (also called teacher models) becomes an interesting topic.

In order to maintain the low-resource and scalability of the model while utilizing the existing tenant knowledge, we propose an algorithm called AdapterDistillation. In AdapterDistillation, we employ adapters to capture task specific features by adding a few extra parameters in the transformer layer, and then distillate knowledge from the existing teacher adapters into the current student adapter. To be exact, when a new tenant comes, we first train an adapter module based on its own local data. Then we load the adapter modules of all the current teacher adapters to assist the training of this new student tenant through knowledge distillation.

Our contributions: The proposed approach has several advantages: 1) Fusion is only added during the second stage of training to guide the current student adapter learning and is not required during inference, ensuring structure consistency between the student adapter and the existing teacher adapters. 2) Since the adapter structure is consistent during prediction and no additional parameters are required, the scalability and low-resource nature of the model itself are retained. To summarize, our contributions are:

- We formulate the construction of a platform-based multi-task problem as a transfer learning problem and leverage the low-resourced adapter model to handle this.
- We propose an AdapterDistillation algorithm which guarantees low-resources and scalability while utilizing the existing tenant knowledge to improve performance.
- We verify noteworthy enhancements of the proposed AdapterDistillation algorithm in terms of accuracy, resource consumption and inference time, using a Frequently Asked Question (FAQ) retrieval service in task-oriented dialog systems.

2 Relevant Work

Recently, adapters have been proposed to capture task-specific features while maintaining similar results to fine-tuning all parameters, which has been widely applied to downstream tasks such as machine translation and cross-lingual transfer (Houlsby et al., 2019; Pfeiffer et al., 2020b; Li and Liang, 2021; Lester et al., 2021; He et al., 2022). Specifically, adapters insert two bottleneck modules into each transformer layer of the pre-trained model (Houlsby et al., 2019). During training, all parameters of the pre-trained model are frozen, and only the parameters of the newly added modules are trained. Some researchers (Pfeiffer et al., 2020a,b) has further improved the insertion position of adapters through structural search, reducing the number of adapter insertions and thus minimizing the increase in parameter quantity and inference speed. A new type of adapters called Lora (Hu et al., 2022) has been proposed to first perform low-rank decomposition on the model parameters and then insert adapters into the key, query, and value matrices of each attention layer. This approach enhances performance and enables parallel execution of the adapter module, thus reducing inference time. Due to the lack of clarity regarding the inter-dependencies and key success factors of various adapter methods, He et al. (He et al., 2022) dissected the design of several classic adapter algorithms and established a unified framework to explore the connections between different adapter methods. Note that all of the work discussed in this paragraph is complimentary to the proposed method called AdapterDistillation since our algorithm is not limited to a certain type of adapters. Thus we can combine our developed approach with all of the work discussed in this paragraph to obtain extra gains.

In addition to optimizing the structure and position of adapters for each individual task, adding extra components on the top of multiple adapters to maximize the transfer of knowledge across multiple tasks is another efficient way to enhance the performance on each task. For example, via adding an extra fusion layer, the AdapterFusion method is proposed to effectively share knowledge across multiple tasks while balancing the various target tasks (Pfeiffer et al., 2021). To be specific, this method uses a two-stage training approach: first, train the corresponding adapter for each task, then load all adapters simultaneously and freeze them, and train an additional adapter fusion layer to aggregate the outputs of all adapters, allowing the model to implicitly and automatically learn to utilize knowledge from different tasks. But this approach faces some challenges in practical applications: since the parameter size of the fusion layer is linearly related to the number of loaded adapters, when the number of adapters is too large, a lot of resources will be used for fusion such that the purpose of using adapters gets lost. Additionally, adding a fusion layer after the adapters leads to larger inference time, resulting in a worse user experience.

To efficiently utilize existing task knowledge and meet the requirement of the platform for streaming task integration, we propose a plug-and-play AdapterDistillation algorithm. By fusing and distilling the knowledge of existing tasks into the current task during training, we can keep the model structure and inference speed unchanged while achieving comparable results to AdapterFusion.

3 Problem Definition

As we know, training adapters for N tasks in parallel might not be practical since tenants often register on the platform in a streaming manner. Based on this fact, the time for the j-th task registered on the platform is assumed to be earlier than that for the i-th task, that is $t_j < t_i$, when $j < i$ for an ordered collection of N tasks denoted as $T = \{t_1, t_2, ..., t_N\}$.

Throughout the paper, we have the following settings which are typically true in practice:

1. The task considered in this paper is non-destuctive, which means when the N-th task is registered on the platform, the performance of the previous $(N-1)$ tasks $\{t_1, t_2, ..., t_{N-1}\}$ should not be impacted.

2. Since the platform often has limited resources, it is reasonable to assume every task needs to be solved with limited computing and memory resources.

3. Due to privacy and security issues, corpus of labeled text for the N-th task is only available locally.

Based on the above setting, in this paper we are aimed at maximizing the transfer of knowledge from the existing tasks to the current new task without impacting the existing tasks, which is more suitable for a practical scenario where each task is registered on the platform in a streaming manner.

4 AdapterDistillation

AdapterFusion allows sharing of information between different tasks through an extra fusion layer at the cost of longer inference time and larger fusion layer (Pfeiffer et al., 2021). However, as a new task is registered on the platform, the existing tasks will be impacted in. In order to mitigate this, we propose AdapterDistillation to allow sharing of information from the existing tasks to the new one while avoiding the impact of the existing tasks without increasing inference time.

4.1 Adapter Learning and Distillation Algorithm

The proposed AdapterDistillation algorithm is a two-stage learning algorithm. In the first stage, we train an adapter model ϕ_N^{first} for the N-th new task when it is registered on the platform based on its own local data.

In the second stage, we employ knowledge distillation to transfer knowledge from the existing tasks to the new adapter, which means the parameter weight of this new adapter will be updated in the second stage. To be exact, assuming there have been $(N-1)$ adapters registered on the platform with their weights being denoted as $\{\phi_n\}_{n=1}^{N-1}$ and the N-th adapter with its weight trained in the first stage being denoted as ϕ_N^{first}. With a fixed pretrained Bert-based model Θ and the existing adapters $\{\phi_n\}_{n=1}^{N-1}$ and ϕ_N^{first}, the data fusion related parameters Ω and the N-th adapter weight ϕ have been introduced to learn how to distill knowledge from the existing adapters $\{\phi_n\}_{n=1}^{N-1}$ and ϕ_N^{first} to better solve the N-th task. The training process can be represented as

$$\Omega_N, \phi_N \leftarrow \arg\min_{\Omega,\phi} CrossEntropy(D_N; \phi, \Theta)$$
$$+\eta \cdot Distill(D_N; \{\phi_n\}_{n=1}^{N-1}, \phi_N^{first}, \Omega, \phi, \Theta) \quad (1)$$

where D_N are corpus of labeled text for task N, η is a predefined constant to balance the distillation loss and the binary cross entropy loss, Ω_N is a set of newly learned fusion-related parameters to transfer the existing knowledge from the existing adapters to the N-th adapter for task N, and ϕ_N is the final weight for adapter N. It is worth mentioning that similar to AdapterFusion, each adapter in AdapterDistillation will be trained twice where the second stage is mainly aimed at implementing knowledge composition. However, different from AdapterFusion which keeps the fusion layer during the inference time, AdapterDistillation will only employ the N-th adapter module to do inference without adding an extra fusion layer (shown in Figure. 2) which leads to faster inference time without impacting the performance of the existing tasks. This makes sense since the N-th adapter weight ϕ_N already contains the sharing of information between N tasks after knowledge distillation.

4.2 Detailed Components

During the training process, AdapterDistillation learns to distill the knowledge from the existing $(N-1)$ adapters to the N-th adapter by introducing the fusion weights Ω and updating the N-th adapter weights ϕ. The fusion weights Ω transfer the existing knowledge to the N-th adapter module by dynamically introducing a distillation loss term as shown in (1). This will push the N-th adapter to learn knowledge not only from its own task data D_N but also from the previous $(N-1)$ adapter intermediate representations.

As shown in Figure 2, our AdapterDistillation architecture contains three components, that is, an adapter fusion, N teacher adapters and a N-th student adapter. In the student adapter part, the output of the feed-forward sublayer of layer l at iteration t, denoted as $\mathbf{h}_{l,t}$, is fed into the N-th adapter to obtain the N-th adapter output $\mathbf{z}_{l,t,N} = g(\mathbf{h}_{l,t}, \phi)$ with $g(\mathbf{h}_{l,t}, \phi)$ being the nonlinear transformation and ϕ being the adapter parameters to be optimized. Interestingly enough, in the N teacher adapters, we not only use the previous $(N-1)$ fully trained adapters ϕ_n as teacher adapters to enable sharing of information between different tasks but also add the N-th partially trained adapter ϕ_N^{first} obtained in the first stage as a teacher adapter to insert some task specific knowledge. In other words, the output of N teacher adapters can be denoted as $\mathbf{z}_{l,t,n} = g(\mathbf{h}_{l,t}, \phi_n)$ for $n = 1, 2, \ldots, N-1$ and $\mathbf{z}_{l,t,N} = g(\mathbf{h}_{l,t}, \phi_N^{first})$.

Similar to AdapterFusion (Pfeiffer et al., 2021), our AdapterDistillation dynamically combines different adapters by introducing Query $\mathbf{Q}_l$, Key $\mathbf{K}_l$, and Value $\mathbf{V}_l$ at each transformer layer l with its complete set being $\Omega = \{\mathbf{Q}_l, \mathbf{K}_l, \mathbf{V}_l\}_{l=1}^{L}$. We employ $\mathbf{z}_{l,t,n}$ for $n = 1, 2, ..., N$ as the input to the value and key transformation to obtain $\mathbf{z}_{l,t,n}^{v} = \mathbf{z}_{l,t,n}^{\top}\mathbf{V}_l$ and $\mathbf{z}_{l,t,n}^{k} = \mathbf{z}_{l,t,n}^{\top}\mathbf{K}_l$, respectively. The output of the feed-forward sublayer $\mathbf{h}_{l,t}$ is used as input to the query transformation to obtain $\mathbf{h}_{l,t}^{Q} = \mathbf{h}_{l,t}^{\top}\mathbf{Q}_l$. Then the output of the adapter fusion $\mathbf{o}_{l,t}$ can be obtained as

$$\mathbf{o}_{l,t} = p_{l,t}^{T}\mathbf{Z}_{l,t,n}^{v} \tag{2}$$

with $p_{l,t} = \text{softmax}(\mathbf{h}_{l,t}^{Q} \bigotimes \mathbf{z}_{l,t,n}^{k})$ and $\mathbf{Z}_{l,t,n}^{v} =$

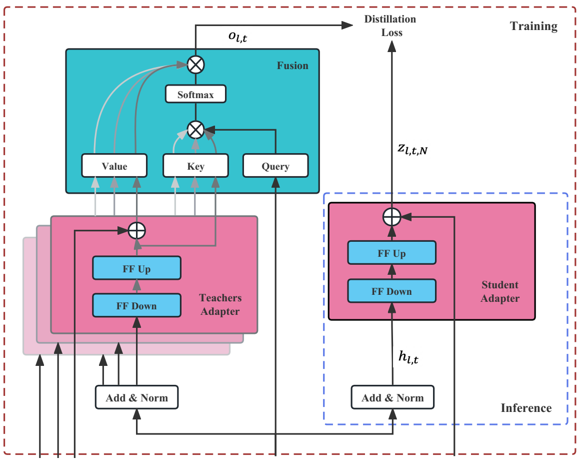

Figure 2: Our AdapterDistillation architecture. This includes trainable weights Query $\mathbf{Q}_l$, Key $\mathbf{K}_l$, Value $\mathbf{V}_l$ and the N-th adapter weight ϕ_N at each transformer layer l.

$[\mathbf{z}_{l,t,1}^{v}, \mathbf{z}_{l,t,2}^{v}, ..., \mathbf{z}_{l,t,N}^{v}]$. Note that we employ $p_{l,t}$ to learn to weight the adapters with regard to the context. Finally, the distillation loss described in (1) is defined as

$$Distill = \|\mathbf{o}_{l,t} - \mathbf{z}_{l,t,N}\|. \tag{3}$$

It is worth mentioning that in the second stage we jointly optimize the adapter fusion Ω and ϕ so as to obtain the optimal ϕ_N which contains the most useful mixed knowledge from available adapters. Then during the inference stage, we only employ ϕ_N to implement the prediction for the N-th task without considering the adapter fusion part so as to reduce inference time. On the other hand, only employing ϕ_N can also lead to comparable performance as AdapterFusion, which will be shown next.

5 Experiments

To validate the effectiveness of AdapterDistillation in terms of accuracy, resource consumption and inference time, its performance is evaluated through a practical scenario where Frequently Asked Question (FAQ) retrieval is considered in task-oriented dialog systems.

5.1 Experimental Setting

To benchmark AdapterDistillation, we compare with the following four model architectures, namely, BERT + adapters (abbr. as Adapter), fully fine-tuning BERT model (abbr. as Full), head-only fine-tuning BERT model (abbr. as HEAD) and AdapterFusion. A detailed experimental setting can be found in Appendix A.1.

Dataset	Full	HEAD	Adapter	AdapterFusion	AdapterDistill
International Payments	76.4(0.859)	64.2(0.714)	74.8(0.834)	76.2(0.855)	75.3(0.84)
Merchant Payments	88.74(0.953)	84.03(0.911)	84.71(0.924)	85.21(0.936)	85.21(0.928)
Broadband Installation	100(1.0)	94.74(0.995)	96.49(0.997)	98.25(1.0)	96.49(0.997)
Cross-border Payments	82.4(0.885)	68.7(0.763)	77.3(0.849)	79.9(0.875)	77.6(0.851)
Merchant Signing	80.39(0.894)	80.39(0.887)	82.35(0.894)	76.47(0.902)	84.31(0.907)
Human Resources	87.02(0.919)	68.11(0.685)	75.17(0.808)	81.32(0.869)	79.73(0.825)
IT Support	99.28(0.999)	76.2(0.855)	93.51(0.982)	96.88(0.993)	94.95(0.988)
Administration	95.77(0.984)	71.96(0.817)	93.65(0.975)	92.59(0.976)	94.71(0.976)
Average	88.75(0.937)	76.04(0.828)	84.75(0.908)	85.85(0.926)	**86.04(0.914)**
Added Params Per Task	100%	0.01%	1.45%	21.36%	**1.45%**

Table 1: The accuracy and AUC for the 10-th tenant with different architectural setups. The result within the parenthesis is AUC. Added Params Per Task represents the percentage of additional parameters added for each task.

5.2 Datasets and Metrics

We select 9 existing tenant models from the platform as teacher adapters, covering fields such as medical care, transportation, insurance, shopping, photography, lease and et al, and employ the performance of the 10-th tenant (student adapter) to evaluate the considered models. In order to reduce the variance, we independently choose 8 unregistered tenants from different business domains as the 10-th student tenant. The 8 independent student tenants are from international payments, merchant payments, broadband installation, cross-border payments, merchant signing, human resources, administrative management, and IT support. The data size for each student tenant ranges from 1000 to 5000 which has been divided into the training, validation, and test dataset with the ratio being 8:1:1. It is worth mentioning that we not only use accuracy and AUC to evaluate the performance, but also use the resource consumption and inference time as additional metrics to indicate the functionality of the models of interest for online practical applications.

5.3 Performance

As shown in Table 1, it is straightforward to see that Full fine-tuning leads to much better performance compared to training only the HEAD (12.71% accuracy increase) at the cost of adding more trainable parameters. Additionally, adapters achieve a little worse accuracy performance compared to Full fine-tuning but with only the 1.45% extra added parameters, which is promising. Table 1 also shows that AdapterFusion can reduce the performance gap by adding an extra fusion layer to implement knowledge composition but at the cost of adding 21.36% extra parameters. Interestingly, the overall accuracy of the propose AdapterDistillation method achieves even better accuracy than AdapterFusion but with only much fewer added parameters (21.36% VS 1.45%). This makes sense since the representations from several such teacher adapters have been inserted into the student adapter through knowledge distillation, which means the fusion layer is not that important for AdapterDistillation during the inference stage.

Storage Space	BERT Fine-Tune	Adapter Fusion	Adapter Distill
500MB	1	0	**18**
1GB	2	6	**109**
5GB	13	53	**815**
10GB	26	111	**1698**
50GB	130	578	**8760**
100GB	261	1161	**17587**

Table 2: The number of tenants that can be supported by different methods versus the storage space.

In addition to accuracy and AUC, resource consumption is an important indicator of deployment costs. In terms of storage space required during the inference stage, the pre-trained Bert-base-Chinese model takes up approximately 391MB. The fusion module and the adapter module occupies 82MB and 3.5MB, respectively. The last classification layer requires 2.3MB. As a result, in addition to the

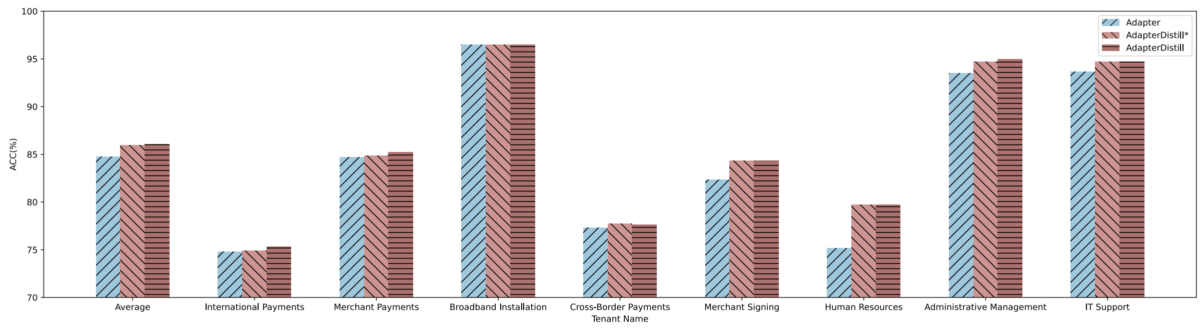

Figure 3: Accuracy of the 10-th tenant from 8 different domains for the different architectural setups. AdapterDistill* is the algorithm where we remove the current tenant as a teacher adapter in the second stage.

Batch Size	BERT Fine-Tune	Adapter Fusion	Adapter/ AdapterDistill
10	25.0	67.6(+170.4%)	25.6(**+2.4%**)
20	43.2	105.8(+144.9%)	44.1(**+2.1%**)
30	65.7	164.2(+149.9%)	67.4(**+2.6%**)

Table 3: Inference of a single forward pass measured in milliseconds, averaged over 100 times.

base model, it takes approximately extra 119.3MB for AdapterFusion but only takes approximately extra 5.8MB for AdapterDistillation when the 10-th tenant registers on the platform. In Table 2, we consider the number of tenants that can be supported by different methods. It shows that when the storage space is 100GB, AdapterDistillation can support 67 times more tenants than Full fine-tuning and 20 times more tenants than AdapterFusion. The results in Table 2 indicate that AdapterDistillation has significant advantages on resource consumption compared to others, which becomes more pronounced as the storage space becomes larger.

For online applications, inference time is closely related to the actual user experience. Next we compare inference time of three algorithms versus different batch sizes. The results in Table 3 show that AdapterDistillation has the same inference time as the Adapter method but it is significantly better than AdapterFusion. This is reasonable because an extra fusion layer in AdapterFusion takes some extra inference time. It is worth noting that the inference time of the Adapter/AdapterDistillation method is slightly larger (about 2.5%) compared to full fine-tuning, which is caused by the serial insertion of the adapter module. Note that AdapterDistillation is independent of the structure of Adapter itself and can be hot-swapped into any Adapter-like method, such as Lora (Hu et al., 2022), to maintain the same inference time as full fine-tuning through parallel insertion.

In order to verify the improvement in model performance is due to the sharing of information from different tasks, we remove the current tenant from teacher adapters and train only using the existing tenants as teacher adapters. Figure 3 indicates that compared to adding the current tenant to the Teachers, the average accuracy is just slightly decreased by about 0.11%, but still outperforms adapters by about 1.18%. This indicates AdapterDistillation can effectively use multiple resources of extracted information.

6 Conclusions and Future Work

We proposed a novel and plug-and-play multi-adapter knowledge distillation algorithm called AdapterDistillation to implement the sharing of information between different tasks. Specifically, our proposed algorithm consisted of two stages of training. In the first stage of training, task-specific knowledge was extracted by training a student adapter using local data. Then in the second stage, knowledge from the existing teacher adapters was distillated into this student adapter through optimizing the distillation loss. Note that AdapterDistillation only employed the trained student adapter to implement inference, which resulted in fast inference time and low resource consumption. Our proposed AdapterDistillation algorithm outperformed existing algorithms in terms of accuracy, resource consumption and inference time, meeting a practical scenario where numerous tenants accessing the platform in a streaming manner. In the future, plugging more advanced adapter structures into AdapterDistillation is an interesting direction to explore.

References

Yicheng Chen, Rick S. Blum, and Brian M. Sadler. 2022a. Communication efficient federated learning via ordered ADMM in a fully decentralized setting. In *56th Annual Conference on Information Sciences and Systems, CISS 2022, Princeton, NJ, USA, March 9-11, 2022*, pages 96–100. IEEE.

Yicheng Chen, Rick S. Blum, Martin Takác, and Brian M. Sadler. 2022b. Distributed learning with sparsified gradient differences. *IEEE J. Sel. Top. Signal Process.*, 16(3):585–600.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM.

Thomas G. Dietterich. 1995. Overfitting and undercomputing in machine learning. *ACM Comput. Surv.*, 27(3):326–327.

Douglas M. Hawkins. 2004. The problem of overfitting. *J. Chem. Inf. Model.*, 44(1):1–12.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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 *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 487–503. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 46–54. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: an adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7654–7673. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Mina Valizadeh and Natalie Parde. 2022. The AI doctor is in: A survey of task-oriented dialogue systems for healthcare applications. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6638–6660. Association for Computational Linguistics.

Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. 2018. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 201–207. Association for Computational Linguistics.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4618–4626. AAAI Press.

A Appendices

A.1 Detailed Experimental Setting

In all experiments, we use Bert-base-Chinese as the pre-training base model and set the classification threshold to be 0.5. All models are trained for 10 epochs with the same learning rate strategy as (Loshchilov and Hutter, 2019). The distillation regularization parameter η in (1) is selected from $[e^{-2}, e^{-1}, e^{0}, e^{1}, e^{2}]$. For AdapterDistillation, we use the same parameter initialization strategy for all key, value, query matrices and the same hyperparameters as AdapterFusion to ensure fair comparison.

A.2 Cold Start and Deployment

Since some new tenants only have a knowledge base without any annotated data at the beginning, a universal pipeline for automatically building tenant datasets is proposed. The knowledge base contains many knowledge points, each of which corresponds to a standard question and multiple similar questions. Note that the collection of questions belonging to the same knowledge point has the same answer.

We automatically construct datasets through the following steps:
1) Download knowledge base with the ID of the newly added tenant.
2) Construct positive samples based on the labeled questions and similar questions in the knowledge base.
3) Constructing negative samples using the BM25 algorithm (Robertson and Zaragoza, 2009).

During the deployment of the service, we load adapter modules for all tenants on the pre-trained model. All requests from tenants on the platform will be directed to this model. During inference, the adapter module belonging to the corresponding tenant is activated based on their name, while those from other tenants are blocked.

https://huggingface.co/bert-base-chinese