Mixture-of-Partitions: Infusing Large Biomedical Knowledge Graphs into BERT

Zaiqiao Meng[♠], **Fangyu Liu**[♠], **Thomas Hikaru Clark**[♠]
Ehsan Shareghi[♣♠], **Nigel Collier**[♠]
[♠]Language Technology Lab, University of Cambridge
[♣]Department of Data Science and AI, Monash University
[♠]{zm324, fl399, thc44, nhc30}@cam.ac.uk
[♣]ehsan.shareghi@monash.edu

Abstract

Infusing factual knowledge into pretrained models is fundamental for many knowledge-intensive tasks. In this paper, we propose **Mixture-of-Partitions (MoP)**, an infusion approach that can handle a very large knowledge graph (KG) by partitioning it into smaller sub-graphs and infusing their specific knowledge into various BERT models using lightweight adapters. To leverage the overall factual knowledge for a target task, these sub-graph adapters are further fine-tuned along with the underlying BERT through a mixture layer. We evaluate our MoP with three biomedical BERTs (SciBERT, BioBERT, PubmedBERT) on six downstream tasks (inc. NLI, QA, Classification), and the results show that our MoP consistently enhances the underlying BERTs in task performance, and achieves new SOTA performances on five evaluated datasets. [1]

1 Introduction

Leveraging factual knowledge to augment pretrained language models is of paramount importance for knowledge-intensive tasks, such as question answering and fact checking (Petroni et al., 2021). Especially in the biomedical domain where public training corpora are limited and noisy, trusted biomedical KGs are crucial for deriving accurate inferences (Li et al., 2020; Liu et al., 2021). However, the infusion of knowledge from real-world biomedical KGs, where entity sets are very large (e.g. UMLS, Bodenreider 2004, contains ∼4M entities) demands highly scalable solutions.

Although many general knowledge-enhanced language models have been proposed, most of them rely on a computationally expensive joint training of an underlying masked language model (MLM) along with a knowledge-infusion objective function to minimize the risk of catastrophic forgetting (Xiong et al., 2019; Zhang et al., 2019; Wang et al., 2021, 2020; Peters et al., 2019; Yuan et al., 2021). Alternatively, entity masking (or entity prediction) has emerged as one of the most popular self-supervised training objectives for infusing entity-level knowledge into pretrained models (Sun et al., 2019; Zhang et al., 2019; Yu et al., 2020; He et al., 2020). However, due to the large number of entities in biomedical KGs, computing an exact softmax over all entities is very expensive for training and predicting (De Cao et al., 2021). Although negative sampling techniques could alleviate the computational issue (Sun et al., 2020), tuning an appropriately hard set of negative instances can be challenging and predicting a very large number of labels may generalize poorly (Hinton et al., 2015).

To address the aforementioned challenges, we propose a novel knowledge infusion approach, named **Mixture-of-Partitions (MoP)**, to infuse factual knowledge based on partitioned KGs into pretrained models (BioBERT, Lee et al. 2020; SciBERT, Beltagy et al. 2019; and PubMedBERT, Gu et al. 2020). More concretely, we first partition a KG into several sub-graphs each containing a disjoint subset of its entities by using the METIS algorithm (Karypis and Kumar, 1998), and then the Transformer ADAPTER module (Houlsby et al., 2019; Pfeiffer et al., 2020b) is applied to learn portable knowledge parameters from each sub-graph. In particular, using ADAPTER module to infuse knowledge does not require fine-tuning the parameters of the underlying BERTs, which is more flexible and efficient while avoiding the catastrophic forgetting issue. To utilise the independently learned knowledge from sub-graph adapters, we introduce mixture layers to automatically route useful knowledge from these adapters to downstream tasks. Figure 1 illustrates our approach.

Our results and analyses indicate that our "divide and conquer" partitioning strategy effectively preserves the rich information presented in two biomedical KGs from UMLS while enabling us to

[1]Our code, models and other related resources can be found in https://github.com/cambridgeltl/mop.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4672–4681
November 7–11, 2021.

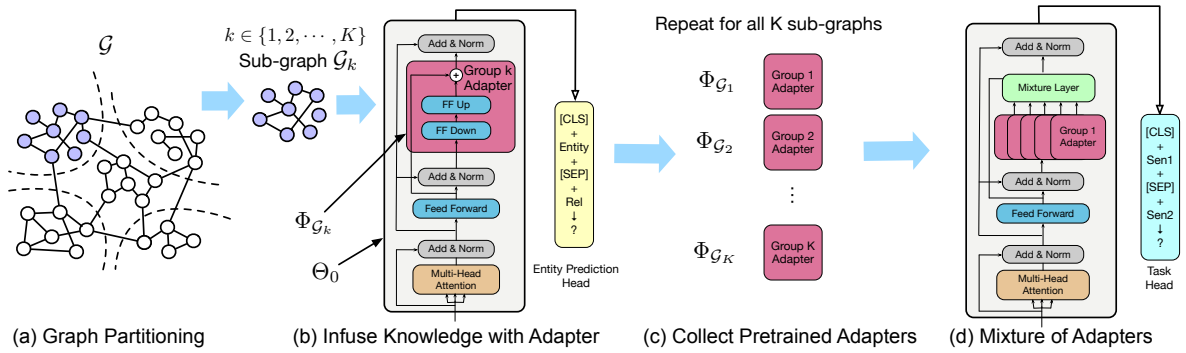

Figure 1: Overview of the proposed MoP.

scale up training on these very large graphs. Additionally, we observe that while individual adapters specialize towards sub-graph specific knowledge, MoP can effectively utilise their individual expertise to enhance the performance of our tested biomedical BERTs on six downstream tasks, where five of them achieve new SOTA performances.

2 Mixture-of-Partitions (MoP)

We denote a KG as a collection of ordered triples $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations, respectively. All the entities and relations are associated with their textual surface forms, which can be a single word (e.g. *fever*), a compound (e.g. *sars-cov-2*), or a short phrase (e.g. *has finding site*).

Given a pretrained model Θ_0, our task is to learn $\Phi_{\mathcal{G}}$ based on an input knowledge graph $\mathcal{G}$, such that it encapsulates the knowledge from $\mathcal{G}$. The training objective, $\mathcal{L}_{\mathcal{G}}$, can be implemented in many ways such as relation classification (Wang et al., 2020), entity linking (Peters et al., 2019), next sentence prediction (Goodwin and Demner-Fushman, 2020), or entity prediction (Sun et al., 2019). In this paper, we focus on entity prediction, one of the most widely used objectives, and leave exploration of other objectives for future work.

As mentioned earlier, exact softmax over all entities is extremely expensive (Mikolov et al., 2013; De Cao et al., 2021) for large-scale KGs, hence we resort to the principle of "divide and conquer", and propose a novel approach called Mixture-of-Partition (MoP). Specifically, our MoP first partitions a large KG into smaller sub-graphs (i.e., $\mathcal{G} \rightarrow \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_K\}$, §2.1), and learns sub-graph specific parameters on each sub-graph separately (i.e., $\{\Phi_{\mathcal{G}_1}, \Phi_{\mathcal{G}_2}, \ldots, \Phi_{\mathcal{G}_K}\}$, §2.2). Then these sub-graph parameters are fine-tuned through mixture layers to route the sub-graph specific knowledge into a target task (§2.3).

2.1 Knowledge Graph Partitioning

Graph partitioning (i.e., partitioning the node set into mutually exclusive groups) is a critical step to our approach, since we need to properly and automatically cluster knowledge triples for supporting data parallelism and controlling computation. In particular, it must satisfy the following goals: (1) maximize the number of resulting knowledge triples to retain as much factual knowledge as possible; (2) balance nodes over partitions to reduce the overall parameters across different entity prediction heads; (3) efficiency at scale for handling large KGs. In fact, an exact solution to (1) and (2) is referred to as the *balanced graph partition* problem, which is NP-complete. We use the METIS (Karypis and Kumar, 1998) algorithm as an approximation, simultaneously meeting all the above three requirements. METIS can handle billion-scale graphs by successively coarsening a large graph into smaller graphs, processing them quickly and then projecting the partitions back onto the larger graph, and has been used in many tasks (Chiang et al., 2019; Defferrard et al., 2016; Zheng et al., 2020).

2.2 Knowledge Infusion with Adapters

Once the large knowledge graph is partitioned, we use ADAPTER modules to infuse the factual knowledge into a pretrained Transformer model by training an entity prediction objective for each sub-graph. ADAPTERs (Houlsby et al., 2019; Pfeiffer et al., 2020b) are newly initialized modules inserted between the Transformer layers of a pretrained model. The training of ADAPTER does not require fine-tuning the existing parameters of the pretrained model. Instead, only the parameters within the ADAPTER modules are updated. In

Model↓, Dataset→	HoC	PubMedQA	BioASQ7b	BioASQ8b	MedQA	MedNLI
SciBERT	$80.52_{\pm 0.60}$	$57.38_{\pm 4.22}$	$75.93_{\pm 4.20}$	$75.72_{\pm 1.79}$	$29.42_{\pm 0.94}$	$81.19_{\pm 0.54}$
+ MoP (SFull)	$81.48^{\dagger}_{\pm 0.35}\uparrow$	$54.78_{\pm 2.96}$	$79.14^{\dagger}_{\pm 3.27}\uparrow$	$74.74_{\pm 1.96}$	$33.03^{\dagger}_{\pm 0.72}\uparrow$	$81.43_{\pm 0.34}\uparrow$
+ MoP (S20Rel)	$81.79^{\dagger}_{\pm 0.66}\uparrow$	$54.66_{\pm 3.10}$	$78.50^{\dagger}_{\pm 4.06}\uparrow$	$76.25_{\pm 2.20}\uparrow$	$32.77^{\dagger}_{\pm 0.67}\uparrow$	$81.20_{\pm 0.37}$
BioBERT	$81.41_{\pm 0.59}$	$60.24_{\pm 2.32}$	$77.50_{\pm 2.92}$	$78.75_{\pm 4.16}$	$30.48_{\pm 0.55}$	$82.42_{\pm 0.59}$
+ MoP (SFull)	$81.47_{\pm 0.89}\uparrow$	$61.82^{\dagger}_{\pm 1.04}\uparrow$	$81.29^{\dagger}_{\pm 3.46}\uparrow$	$82.04^{\dagger}_{\pm 4.59}\uparrow$	$33.55^{\dagger}_{\pm 0.56}\uparrow$	$83.44_{\pm 0.24}\uparrow$
+ MoP (S20Rel)	$82.53^{\dagger}_{\pm 1.08}\uparrow$	$61.04_{\pm 4.81}\uparrow$	$80.79^{\dagger}_{\pm 4.40}\uparrow$	$80.00_{\pm 5.03}\uparrow$	$34.15^{\dagger}_{\pm 0.79}\uparrow$	$82.93_{\pm 0.55}\uparrow$
PubMedBERT	$82.25_{\pm 0.46}$	$55.84_{\pm 1.78}$	$87.71_{\pm 4.25}$	$84.54_{\pm 2.36}$	$35.08_{\pm 0.22}$	$84.18_{\pm 0.19}$
+ MoP (SFull)	$82.71_{\pm 0.56}\uparrow$	$61.74^{\dagger}_{\pm 2.54}\uparrow$	$88.64_{\pm 3.04}\uparrow$	$86.12^{\dagger}_{\pm 2.39}\uparrow$	$36.33^{\dagger}_{\pm 0.16}\uparrow$	$84.25_{\pm 0.25}\uparrow$
+ MoP (S20Rel)	$\mathbf{83.26}^{\dagger}_{\pm 0.32}\uparrow$	$\mathbf{62.84}^{\dagger}_{\pm 2.71}\uparrow$	$\mathbf{90.64}^{\dagger}_{\pm 2.43}\uparrow$	$85.39_{\pm 1.51}\uparrow$	$\mathbf{38.02}^{\dagger}_{\pm 0.05}\uparrow$	$\mathbf{84.70}_{\pm 0.19}\uparrow$
SOTA	82.32 (Gu et al., 2020)	60.24 (Gu et al., 2020)	87.56 (Gu et al., 2020)	**90.32** (Nentidis et al., 2020)	36.70 (Jin et al., 2020)	83.80 (Peng et al., 2019)

Table 1: Performance on various tasks. The best ones are in **bold**, while ↑ denotes that improvements are observed comparing with the base model. "†" denotes statistically significant better than the base model (T-test, $p < 0.05$).

this paper, we use the ADAPTER module configured by Pfeiffer et al. (2020a), which is shown in Figure 1 (b). In particular, given a sub-graph $\mathcal{G}_k$, we remove the tail entity name for each triple $(h, r, t) \in \mathcal{G}_k$, and transform the triple into a list of tokens: '[CLS] h [SEP] r [SEP]'. The sub-graph specific ADAPTER module is trained to predict the tail entity using the representation of the [CLS] token and the parameters $\Phi_{\mathcal{G}_k}$ are optimized by minimizing the cross-entropy loss. During the fine-tuning of downstream tasks, both the parameters of ADAPTER and pre-trained LM will be updated.

2.3 Mixture Layers

Given a set of knowledge-encapsulated adapters, we use AdapterFusion mixture layers to combine knowledge from different adapters for downstream tasks. AdapterFusion is a recently proposed model (Pfeiffer et al., 2020a) that learns to combine the information from a set of task adapters by a softmax attention layer. It learns a contextual mixture weight over adapters at layer l using an attention with the softmax weights:

$$s_{l,k} = \text{Softmax}(\Phi_{l,\mathcal{G}_1}, \Phi_{l,\mathcal{G}_2}, \cdots, \Phi_{l,\mathcal{G}_K}; \Theta_{l,0}), \quad (1)$$

where $s_{l,k}$ is used to mix the adapter outputs to be passed into the next layer, and the final layer L is used to predict a task label y:

$$y = f(\sum_{k=1}^{K} s_{L,k} \Phi_{L,\mathcal{G}_k}), \quad (2)$$

where f is the target task prediction head. Closely related to ours is the sparsely-gated Mixture-of-Experts layer (Shazeer et al., 2017). Alternatively, a more flexible mechanism such as Gumbel-Softmax (Jang et al., 2017) can be used for obtaining more discrete/continuous mixture weights. However, we found both alternatives underperform AdapterFusion (see Appendix for a comparison).

3 Experiments

3.1 Bio-medical Knowledge Graphs

	# Entities	# Relations	# Triples
SFull	302,332	229	4,129,726
S20Rel	263,808	20	1,750,677

Table 2: Statistics of the used two SNOMED-CT KGs.

We evaluate our proposed MoP on two KGs, named **SFull** and **S20Rel**, which are extracted from the large biomedical knowledge graph **UMLS** (Bodenreider, 2004) under the *SNOMED CT, US Edition* vocabulary. The **SFull** KG contains the full relations and entities of SNOMED[2], while the **S20Rel** KG is a sub-set of **SFull** that only contains the top 20 most frequent relations. Note that since some relations in **SFull** are the reversed mappings of the same entity pairs, e.g. "A has causative agent B" and "B causative agent of A", therefore for **S20Rel** we exclude those reversed relations in the top 20 relations. Table 2 shows the statistics of the two KGs and the used 20 relations of the **S20Rel** are listed in the appendix.

3.2 Evaluated Tasks and Datasets

We evaluate our MoP on six datasets over various downstream tasks, including four question answering (i.e., PubMedQA, Jin et al. 2019; BioAsq7b, Nentidis et al. 2019; BioAsq8b, Nentidis et al. 2020; MedQA, Jin et al. 2020), one document classification (HoC, Baker and Korhonen 2017), and one natural language inference (MedNLI, Romanov and Shivade 2018) datasets. While HoC is a multi-label classification, and MedQA is a multi-choice prediction, the rest can be formulated as

[2]https://www.snomed.org/

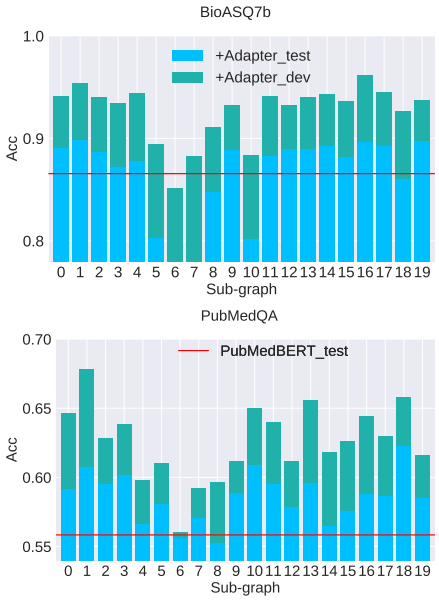

Figure 2: Performances of adapters over 20 partitions.

a binary/multiclass classification tasks. See Appendix for the detailed description of these tasks and their datasets.

3.3 Pretraining with Base Models

We experiment with three biomedical pretrained models, namely BioBERT (Lee et al., 2020), SciBERT (Beltagy et al., 2019) and PubMedBERT (Gu et al., 2020), as our base models, which have shown strong progress in biomedical text mining tasks. We first partition our KGs into different number of sub-graphs (i.e. $\{5, 10, 20, 40\}$), then for each sub-graph, we train the base models loaded with the newly initialized ADAPTER modules (with a compression rate CRate $= 8$) for 1-2 epochs by minimizing the cross-entropy loss. AdamW (Loshchilov and Hutter, 2018) is used as our training optimizer, and the learning rates for all the sub-graphs are fixed to $1e-4$, as suggested by (Pfeiffer et al., 2020b). Unless specified otherwise, all the reported performances are based on a partition of 20 sub-graphs, since this was optimal for task performance (see Section 3.7 for the performances over different number of partitions.).

3.4 Partition Evaluation on Tasks

In Figure 2 we report the average performance (10 runs) of the knowledge-infused PubMedBERT on two QA datasets over partitioned **SFull**. We can see that partitions contribute to various degrees while some (e.g. #5) have a negligible benefit. However, the role of the least contributing partitions could not be discarded as we repeated our downstream tasks by keeping only the top-10 performing partitions and observed the results were still worse than the model trained on all 20 partitions (i.e., accuracy drop of 2.7 on PubMedQA, and 1.4 on BioASQ7b).[3] This highlights the importance of automatically learning the contribution weights of partitions.

Ratio	SFull	S20Rel
0%	2,830,674 (100.0%)	1,225,708 (100.0%)
10%	1,619,278 (57.2%)	990,417 (80.8%)
20%	1,302,096 (46.0%)	789,220 (64.4%)
40%	805,802 (28.5%)	479,807 (39.1%)
80%	289,886 (10.2%)	156,374 (12.8%)
100%	250,914 (8.9%)	114,695 (9.4%)

Table 3: Number of training triples numbers over different shuffling ratios. With 0% shuffling rate indicating the METIS 20-partitioned sub-graphs, and 100% being a totally randomly generated partitions.

3.5 MoP Evaluation on Tasks

Table 1 shows the overall performance of our MoP deployed on SciBERT, BioBERT and PubMedBERT pretrained models. We see that MoP pretrained on the SFull KG improves both the BioBERT and PubMedBERT models for all the tasks, while the SciBERT model can be also improved on 4 out of 6 tasks. The result shows that MoP pretrained with the S20Rel KG achieves new SOTA performances on four tasks. This suggests further pruning of the knowledge triples helps task performance by reducing noise, and is a promising direction to explore in future.

3.6 METIS Partitioning Quality

We design a controlled random partitioning scheme to test whether METIS can produce high quality partitions for training. We fix the entity size for a 20-partitioned result produced by METIS, and randomly shuffle a percentage (ranging from 0%-100%) of entities across all the sub-graphs. Table 3 shows the number of training triples numbers over different shuffling ratios. In Figure 3 we report the results on BioASQ7b and PubMedQA under different shuffling rates. We can see that the performances of MoP on both datasets degrades significantly as the shuffling rate increases, which highlights the quality of the produced partitions.

[3] See Appendix for performances on the split sub-graphs.

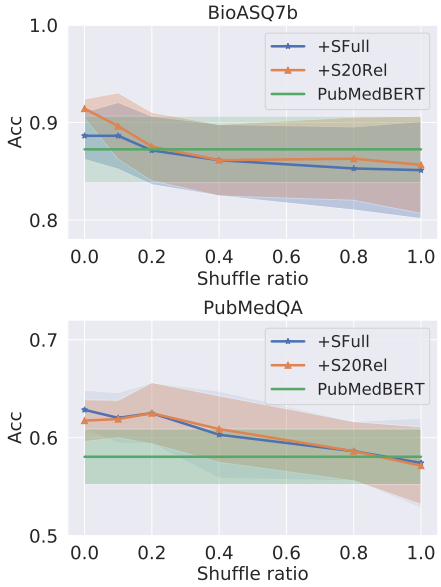

Figure 3: Performance vs. shuffling rates. The shaded regions are the standard deviations (20 runs).

# Sub-graphs (Avg. # entity)	BioASQ7b	PubMedQA
5 (60,466)	86.61	59.22
10 (30,233)	87.86	60.38
20 (15,116)	**88.64**	**61.74**
40 (7,558)	86.54	58.70
60 (5,038)	87.11	59.48

Table 4: Accuracy performance of MoP over different number of partitions.

3.7 Performance vs. Number of Partitions

Table 4 shows the performance of PubMedBERT+MoP trained on the SFull knowledge graph over different number of partitions. We can clearly see that under 20 partitions, PubMedBERT+MoP performs the best in both of the BioASQ7b and PubMedQA datasets, and an average entity size of 15k-30k for the sub-graphs usually yields better performance than others.

3.8 Case Study

In Figure 4, we show six examples (contexts are omitted for brevity) from BioASQ8b and compare their mixture weights of the final layer [CLS] token inferred by the PubMedBERT+MoP (SFull) model. We see that each question elicits different mixture weights, indicating that MoP can leverage the expertise of different sub-graphs depending on the target example. We also plot the word cloud over six groups of sub-graphs that are clustered by k-means according to the entity name's TF-IDF

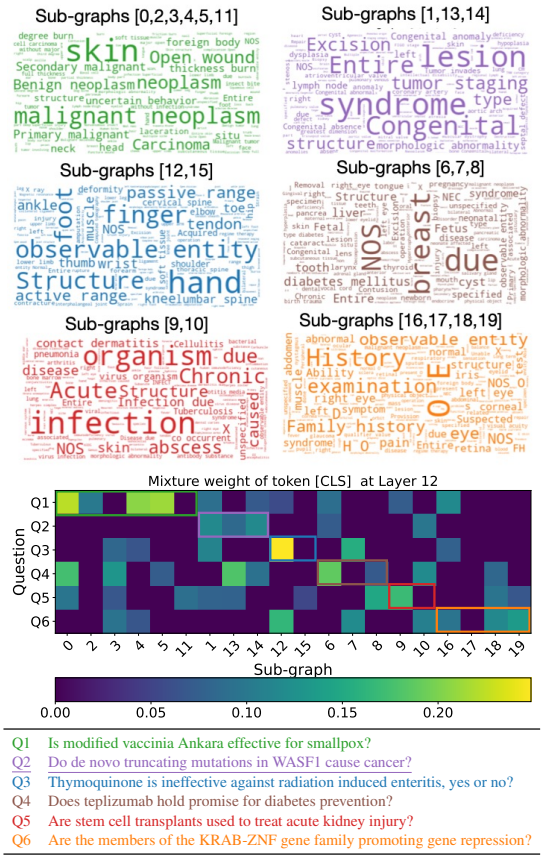

Figure 4: **Top**: word cloud over 6 groups clustered from 20 sub-graphs. **Middle**: mixtures weights of MoP on 6 questions. **Bottom**: 6 questions from BioASQ8b.

feature of these sub-graphs. We can observe that MoP identifies the most related sub-graphs for each example (e.g. Q2 has more weights on sub-graphs [1,13,14], which specialise in 'tumor' knowledge). This validates the effectiveness of our MoP in balancing useful knowledge across adapters.

4 Conclusion and Future Work

In this paper, we proposed **MoP**, a novel approach for infusing knowledge by partitioning knowledge graphs into smaller sub-graphs. We show that while the knowledge-encapsulated adapters perform very differently over different sub-graphs, our proposed MoP can automatically leverage and balance the useful knowledge across those adapters to enhance various downstream tasks. In the future, we will evaluate our approach using some general domain KGs based on some general domain tasks.

Acknowledgements

Nigel Collier and Zaiqiao Meng kindly acknowledge grant-in-aid funding from ESRC (grant number ES/T012277/1).

References

Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. *BioNLP 2017*, pages 307–315.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP-IJCNLP*, pages 3606–3611.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *SIGKDD*, pages 257–266.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pages 3844–3852.

Travis R Goodwin and Dina Demner-Fushman. 2020. Enhancing question answering by injecting ontological knowledge through regularization. In *EMNLP*, volume 2020, page 56.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779*.

Yun He, Ziwei Zhu, Yin Zhang, Qin Chen, and James Caverlee. 2020. Infusing disease knowledge into BERT for health question answering, medical inference and disease name recognition. In *EMNLP*, pages 4604–4614.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. In *NeurIPS*.

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 *ICML*, pages 2790–2799.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *arXiv preprint arXiv:2009.13081*.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *EMNLP-IJCNLP*, pages 2567–2577.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Dongfang Li, Baotian Hu, Qingcai Chen, Weihua Peng, and Anqi Wang. 2020. Towards medical machine reading comprehension with structural knowledge and plain text. In *EMNLP*, pages 1427–1438.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2021. Self-alignment pre-training for biomedical entity representations. *NAACL*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119.

Anastasios Nentidis, Konstantinos Bougiatiotis, Anastasia Krithara, and Georgios Paliouras. 2019. Results of the seventh edition of the BioASQ challenge. In *ECML PKDD*, pages 553–568.

Anastasios Nentidis, Anastasia Krithara, Konstantinos Bougiatiotis, Martin Krallinger, Carlos Rodriguez-Penagos, Marta Villegas, and Georgios Paliouras. 2020. Overview of BioASQ 2020: The eighth BioASQ challenge on large-scale biomedical semantic indexing and question answering. In *CLEF*, pages 194–214.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *EMNLP-IJCNLP*, pages 43–54.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. KILT: a benchmark for knowledge intensive language tasks. *NAACL*.

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

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020b. AdapterHub: A framework for adapting transformers. In *EMNLP*, pages 46–54.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *EMNLP*, pages 1586–1596.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuan-Jing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. In *COLING*, pages 3660–3670.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2019. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *ICLR*.

Donghan Yu, Chenguang Zhu, Yiming Yang, and Michael Zeng. 2020. Jaket: Joint pre-training of knowledge graph and language understanding. *arXiv preprint arXiv:2010.00796*.

Zheng Yuan, Yijia Liu, Chuanqi Tan, Songfang Huang, and Fei Huang. 2021. Improving biomedical pretrained language models with knowledge. *arXiv preprint arXiv:2104.10344*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *ACL*, pages 1441–1451.

Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. 2020. DistDGL: Distributed graph neural network training for billion-scale graphs. *arXiv preprint arXiv:2010.05337*.

Appendix

A.1 The used 20 relations in the S20Rel knowledge graph

ID	Relation
0	has causative agent
1	has active ingredient
2	has direct substance
3	has finding site
4	has direct procedure site
5	has procedure site
6	possibly equivalent to
7	has occurrence
8	has associated morphology
9	has direct morphology
10	interprets
11	has method
12	has direct device
13	has dose form
14	has subject relationship context
15	has pathological process
16	has interpretation
17	moved to
18	has intent
19	has temporal context

Table 5: Relations used in **S20Rel** knowledge graph.

A.2 Evaluated Datasets and Experiment details

We evaluate our MoP on six datasets over various downstream tasks, including four question answering (i.e., PubMedQA, Jin et al. 2019; BioAsq7b, Nentidis et al. 2019; BioAsq8b, Nentidis et al. 2020; MedQA, Jin et al. 2020), one document classification (HoC, Baker and Korhonen 2017), and one natural language inference (MedNLI, Romanov and Shivade 2018) datasets. While HoC is a multi-label classification, and MedQA is a multi-choice prediction, the rest can be formulated as a binary/multiclass classification tasks.

- **HoC** (Baker and Korhonen, 2017): The Hallmarks of Cancer corpus was extracted from 1852 PubMed publication abstracts by Baker and Korhonen (2017), and the class labels were manually annotated by experts according to the Hallmarks of Cancer taxonomy. The taxonomy consists of 37 classes in a hierarchy, but in this paper we only consider the ten top-level ones. We use the publicly available train/dev/test split created by (Gu et al., 2020) and report the average performance over five runs by the average micro F1 across the ten cancer hallmarks.

- **PubMedQA** (Jin et al., 2019): This is a question answering dataset that contains a set of research questions, each with a reference text from a PubMed abstract as well as an annotated label of whether the text contains the answer to the research question (*yes/maybe/no*). We use the original *train/dev/test* split with 450/50/500 questions, respectively. The reported performance are the average of ten runs under the accuracy metric.

- **BioASQ7b, BioASQ8b** (Nentidis et al., 2019, 2020): The both BioASQ datasets are *yes/no* question answering tasks annotated by biomedical experts. Each question is paired with a reference text containing multiple sentences from a PubMed abstract and a *yes/no* answer. We use the official *train/dev/test* splits, i.e. 670/75/140 and 729/152/152 for BioASQ7b and BioASQ8b respectively, and the reported performances are the average of ten runs under the accuracy metric.

- **MedNLI** (Romanov and Shivade, 2018): MedNLI is a Natural Language Inference (NLI) collection of sentence pairs extracted from MIMIC-III, a large clinical database. The objective of the NLI task is to determine if a given hypothesis can be inferred from a given premise. This task is formulated as the document classification task over three labels: {entailment, contradiction,neutral}. We use the same train/dev/test split generated by Romanov and Shivade (2018), and report the average accuracy performance over three runs.

- **MedQA** (Jin et al., 2020): MedQA is a publicly available large-scale multiple-choice question answering dataset extracted from the professional medical board exams. It covers three languages: English, simplified Chinese, and traditional Chinese, but in this paper we only adopt the English set, which is split by Jin et al. (2020). Following (Jin et al., 2020), we use the Elasticsearch system to retrieve the top 25 sentences to each question+choice pair as the context for each choice, and concatenate them to obtain the normalized log probability over the five choices. Since this

dataset is very large, we only report the average accuracy performance under three runs for all the models.

A.3 Comparison of Different Mixture Approaches

Our MoP approach first infuses the factual knowledge for all the partitioned sub-graphs using their respective adapters with newly initialized parameters, then these knowledge-encapsulated adapters are further fine-tuned alone with the underlying BERT model through mixture layers. In this paper, we explored three approaches for implementing the mixture layers, which are described as follows:

- **Softmax.** As the default mixture layers deployed in our MoP, **AdapterFusion** is a recent proposed model (Pfeiffer et al., 2020a) that learns to combine the information from a set of tasks adapters by a softmax attention layer. In particular, the outputs from different adapters at layer l are combined using a contextual mixture weight calculated by a softmax over these adapters:

$$s_{l,k} = \text{Softmax}(\Phi_{l,\mathcal{G}_1}, \Phi_{l,\mathcal{G}_2}, \cdots, \Phi_{l,\mathcal{G}_K}; \Theta_{l,0}) \,. \tag{3}$$

  For brevity, we denote our MoP with the original AdapterFusion mixture layers as **Softmax**.

- **Gumbel.** We also extend the AdapterFusion by replacing the softmax layer with the **Gumbel-Softmax** (Jang et al., 2017) layer for obtaining more discrete mixture weights:

$$\begin{aligned} s_{l,k} &= \text{Gumbel-Softmax}(\Phi_{l,\mathcal{G}_1}, \Phi_{l,\mathcal{G}_2}, \cdots, \Phi_{l,\mathcal{G}_K}; \Theta_{l,0}) \\ &= \frac{\exp\left(\log \Phi_{l,\mathcal{G}_k} + g_k\right)/\tau}{\sum_{i=1}^{K} \exp\left(\log \Phi_{l,\mathcal{G}_i} + g_i\right)/\tau}, \end{aligned} \tag{4}$$

  where $g_1, \cdots, g_K$ are i.i.d samples drawn from $Gumbel(0, 1)$ distribution, and τ is a hyper-parameter controlling the discreteness. For brevity, we denote our MoP with the Gumbel-Softmax AdapterFusion mixture layers as **Gumbel**.

- **MoE. Mix-of-Experts (MoE)** is a type of general purpose neural network component for selecting a combination of the experts to process each input. In particular, we use the the sparsely-gated mixture-of-experts, introduced by Shazeer et al. (2017), for obtaining a top-K sparse mixture of these adapters. And the mixture weights are calculated by:

$$s_{l,k} = \text{Softmax}(\text{TopK}(H(\Phi_{l,\mathcal{G}_k}), K)), \tag{5}$$

  where $H(\Phi_{l,\mathcal{G}_k})$ is a function for transferring hidden variables into scalars with tunable Gaussian noise, and $\text{TopK}(\cdot)$ is a function for keeping only the top K values. We denote our MoP with this mixture approach as **MoE**.

Model	BioASQ7b	PubMedQA
Gumbel ($\tau = 0.1$)	85.21±7.5	59.90±2.8
Gumbel ($\tau = 0.25$)	87.07±2.7	59.94±1.8
Gumbel ($\tau = 0.5$)	87.07±3.8	59.46±2.1
Gumbel ($\tau = 0.75$)	87.14±3.6	59.35±1.9
MoE ($K = 2$)	86.93±6.5	60.06±5.1
MoE ($K = 3$)	87.36±6.2	60.26±4.5
MoE ($K = 5$)	86.79±4.5	59.32±4.4
MoE ($K = 10$)	85.57±4.8	59.02±5.7
MoE ($K = 15$)	88.29±2.3	61.52±4.9
Softmax	**88.64**±3.0	**61.74**±2.7

Table 6: Performance comparison on BioASQ7b and PubMedQA tasks over the three mixture approaches.

Table 6 shows the performance comparison of the three mixture approaches on the BioASQ7b and PubMedQA tasks. Note that Gumbel and MoE have additional hyper-parameters, i.e. τ and K, for controlling the discreteness and topK respectively. From Table 6, we can see that the original AdapterFusion with the softmax layer outperforms other two mixture approaches on all the hyper-parameter choices. This result justifies the choice of the mixture layers in our MoP model.

A.4 Performance on the Split Sub-graphs

To further validate that the performance improvements of the evaluated BERTs using our MoP are gained due to the infused knowledge from the sub-graph adapters, rather than the newly added more parameters of adapters, we split the partitioned sub-graphs into two groups according to their test performance ranking, and use our MoP to fine-tune the adapters of each group. Table 7 and 8 show the performances of all the adapters and our MoP combining the grouped adapters over train/dev/test sets of the BioASQ7b and PubMedQA datasets respectively. As we can see from the two tables, our MoP fine-tuned under the group of higher performance adapters can consistently obtain better performance than the group of the lower performance adapters. Note that we have shown that in Figure 2 adapters pretrained by different sub-graphs show quite different performances, and each sub-graph adapter can capture different factual knowledge. Then, the result from Tab 7 and 8 further validates that the marginal performance gains of our MoP are indeed

caused by the mixture of knowledge infused from adapters.

Sub-graph	Adapters			MoP		
	train	dev	test	train	dev	test
6	0.892	0.852	0.735			
7	0.969	0.883	0.780			
10	0.936	0.884	0.802			
5	0.961	0.895	0.804			
8	0.989	0.911	0.848	0.992	0.9467	0.8619
18	0.980	0.927	0.861			
3	0.998	0.935	0.873			
4	0.998	0.944	0.878			
15	0.993	0.936	0.882			
11	0.979	0.941	0.883			
2	0.999	0.940	0.887			
9	0.992	0.932	0.889			
13	0.995	0.940	0.889			
12	0.994	0.932	0.890			
0	0.997	0.941	0.891	0.997	0.9467	0.9
14	0.998	0.943	0.893			
17	0.993	0.945	0.893			
16	0.997	0.961	0.896			
19	0.991	0.937	0.898			
1	0.998	0.953	0.899			

Table 7: MoP Performance on BioASQ7b over two groups of sub-graphs, which are divided according to their test accuracy performance.

Sub-graph	Adapters			MoP		
	train	dev	test	train	dev	test
8	0.739	0.596	0.553			
6	0.606	0.560	0.556			
14	0.823	0.618	0.565			
4	0.732	0.598	0.567			
7	0.707	0.592	0.571	0.8409	0.648	0.5824
15	0.835	0.626	0.575			
12	0.817	0.612	0.579			
5	0.748	0.610	0.581			
19	0.845	0.616	0.585			
17	0.912	0.630	0.587			
16	0.909	0.644	0.588			
9	0.797	0.612	0.589			
0	0.864	0.646	0.592			
2	0.854	0.628	0.595			
11	0.935	0.640	0.595	0.982	0.668	0.6006
13	0.906	0.656	0.597			
3	0.867	0.638	0.602			
1	0.906	0.678	0.607			
10	0.816	0.650	0.609			
18	0.794	0.658	0.623			

Table 8: MoP Performance on PubMedQA over two groups of sub-graphs, which are divided according to their test accuracy performance.