A Technical Question Answering System with Transfer Learning

Wenhao Yu†, Lingfei Wu‡, Yu Deng‡, Ruchi Mahindru‡,
Qingkai Zeng†, Sinem Guven‡, Meng Jiang†
†University of Notre Dame, Notre Dame, IN, USA
‡IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA
†{wyu1, qzeng, mjiang2}@nd.edu
‡{wuli, dengy, rmahindr, sguven}@us.ibm.com

Abstract

In recent years, the need for community technical question-answering sites has increased significantly. However, it is often expensive for human experts to provide timely and helpful responses on those forums. We develop TransTQA, which is a novel system that offers automatic responses by retrieving proper answers based on correctly answered similar questions in the past. TransTQA is built upon a siamese ALBERT network, which enables it to respond quickly and accurately. Furthermore, TransTQA adopts a standard deep transfer learning strategy to improve its capability of supporting multiple technical domains.

1 Introduction

Technical community (e.g., StackOverflow) is the most active and compelling open forum community for question answering (CQA) that has played an important role for technicians sharing and spreading professional knowledge (Srba and Bielikova, 2016). Recently, many technology companies have been developing and actively maintaining their own technical support forums (e.g., IBM DeveloperWorks) where users/consumers can post technical issues to seek advice and solutions from peers and experts. Technology companies are investing in training their employees with professional communication skills and technical knowledge to effectively respond to users' questions. However, it is often expensive to provide timely and helpful responses on these forums. Statistics show that at least 30% questions do not hold an accepted answer on StackOverflow and AskUbuntu. First, the users/peers often have limited domain-specific knowledge to describe their issues or accurately answer complex questions in appropriate technical terminology. Second, it is common for such forum threads to become multi-turn asynchronous interactions, such that the users/peers give additional information, forming a series of discussions, which is time-consuming for experts to track such ongoing interactions. Therefore, building an intelligent system to automatically respond to new questions with useful information or a solution is in high demand. In scenarios where the automatic response does not address a user's technical issue, human experts can further intervene such discussions. A straightforward approach is to properly match potential candidate answers to the new question, since many questions have recurred, allowing for new questions to be answered using the historically accepted answers.

The Information Retrieval (IR) and Natural Language Processing (NLP) communities have witnessed the growing dominance of retrieval approaches based on neural networks over the last decade (Abbasiyantaeb and Momtazi, 2020). Recent advances of pre-training language models (e.g., BERT) achieved superior performance on the retrieval Question Answering (QA) task (Reimers and Gurevych, 2019; Chang et al., 2020). Several BERT based QA systems in the general domain have been deployed for real-world applications (Yang et al., 2019; Yilmaz et al., 2019). In these approaches, BERT first concatenates a QA pair as [CLS, Q, SEP, A, SEP], then passes the concatenation to the transformer network (Devlin et al., 2019). This setup is time-consuming and resource intensive for retrieval task because of too many possible combinations. Finding the pair of the highest similarity in a collection of 10,000 sentences requires 49,995,000 inference computations in BERT. On a modern V100 GPU, it requires around 60 hours. Furthermore, while different technical communities have different focuses, there is still substantial overlap in the abilities of intelligent systems required to answer questions across multiple technical domains. To the best of our

knowledge, there is no existing system that uses transfer learning to address technical QA tasks.

We develop a novel system called TransTQA (in short for "**trans**fer learning for **t**echnical **q**uestion **a**nswering"). Our system leverages accepted questions and answers (tagged by human users) for answering similar technical questions. The system consists of two modes: evaluation mode and usage mode. The evaluation mode demonstrates retrieved answers by four different transfer learning strategies. The usage mode displays three highest ranked answers retrieved from the database of answer candidates. We employ the latest algorithm ALBERT (Lan et al., 2020) with a siamese network, to make our system efficient and accurate. First, an ALBERT has 18x fewer parameters than BERT-large with the same configuration. A siamese network has two separate ALBERT encoders to generate question and answer embeddings, and applies a similarity measure at the top layer. The similarity measuring process can perform efficiently on modern hardware, allowing our system to be used for real-time usage. Second, our system demonstrates superior accuracy compared to traditional LSTM based QA systems (Rücklé and Gurevych, 2017; Loginova and Neumann, 2018). Third, we adopt a standard deep transfer learning technique, so our system is able to work on different technical domains. Experiments show our proposed system is both time efficient and memory efficient.

We have provide the following links for readers to easily access our (1) demo video and (2) source code (3) website for research purpose.
(1) Video: `https://vimeo.com/431118548`
(2) Code: `https://github.com/wyu97/TTQA`
(The website URL may change, please go to the above github page to view real-time updates.)

2 Related Work

Question Answering Question answering is a long-standing challenge in NLP. Several QA benchmarks have been introduced over the past decade, such as answer selection (Yang et al., 2015) and reading comprehension (Rajpurkar et al., 2016). As an increasing number of researchers are focusing on the above tasks, the role of retrieval has been gradually overlooked. In real-world scenarios, however, it is less practical to assume people are given a small set of candidate answers or a golden passage. The large database of answers to previous questions should be properly utilized for answering new questions. So we focus on retrieval QA in the task formulation and model architecture.

Retrieval Question Answering Retrieval Question Answering (a.k.a Answer Retrieval) finds the most similar answer between multiple candidate answers for a given question (Abbasiyantaeb and Momtazi, 2020). Recently, several researchers have proposed different deep neural models in text-based QA that compare two segments of texts and produces a similarity score. Both document-level (Chen et al., 2017; Seo et al., 2018, 2019; Wu et al., 2018) and sentence-level (Ahmad et al., 2019; Yu et al., 2020) retrieval has been studied on many public datasets such as SQuAD (Rajpurkar et al., 2016) and NQ (Kwiatkowski et al., 2019).

Transfer Learning for QA Transfer learning studies how to transfer knowledge from a source domain to a target domain (Pan and Yang, 2009; Jiang et al., 2016). Recent advances of deep transfer learning technologies have achieved great success in various NLP tasks (Ruder et al., 2019). Several research work in this domain greatly enrich the application and technology of transfer learning on question answering from different perspectives (Min et al., 2017; Golub et al., 2017; Deng et al., 2018; Wang et al., 2019; Castelli et al., 2020). Although transfer learning has been successfully applied to various QA applications, its applicability to technical QA has yet to be investigated.

Question Answering System Early neural based QA systems (Kato et al., 2017; Loginova and Neumann, 2018; Chen et al., 2019) are often based on QALSTM models (Tan et al., 2016) with self-attention mechanism (Lin et al., 2017) in order to visualize and illuminate the inner workings of a specific LSTM. As recent advances of Transformer (Vaswani et al., 2017) and BERT (Devlin et al., 2019) achieved superior performance on many NLP tasks in general domains, Transformer-based QA systems (Ma et al., 2019) and fine-tuned BERT QA systems (Yang et al., 2019; Yilmaz et al., 2019) have been deployed to better retrieve answers.

3 Proposed System

3.1 System Configuration

Our system is configured with ALBERT model according to the specified arguments that defines the model architecture. The backbone of the ALBERT

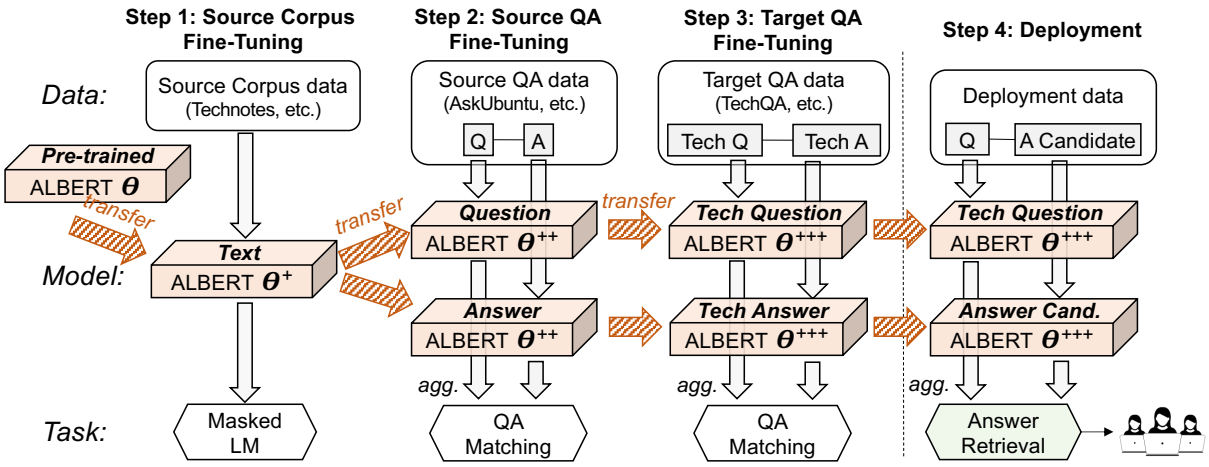

Figure 1: High level architecture of our proposed TransTQA system. First, the pre-trained ALBERT model is fine tuned with unstructured source technical corpus with masked language model (MLM) task, i.e., $\theta \rightarrow \theta^{+}$. Second, a siamese ALBERT takes fine tuned ALBERT and fine tunes with source technical QA, i.e., $\theta^{+} \rightarrow \theta^{++}$. Third, the siamese ALBERT further fine tunes with target QA, i.e., $\theta^{++} \rightarrow \theta^{+++}$. Our deployed system takes θ^{+++}. Given a query, our system first calculates similarity scores between the query and each candidate answer, then ranks all scores from highest to lowest. Finally, the system returns top-3 ranked answers.

architecture is similar to BERT in that it also uses a Transformer encoder. In order to allow reproducability, we integrate Huggingface Transformer (Wolf et al., 2019) into the system construction. Huggingface is an open-source toolkit containing more than 10 popular pre-training language model implementations (e.g., BERT, XLNet). Therefore, when reproducing the system with other data, only the model name would need to be adjusted (e.g., BERT-Large, XLNet-Base) according to the training resources without manually adjust the specified model configurations. Different pre-training language models can be found at HuggingFace[1]. For instance, the configurations of our siamese-ALBERT are automatically taken as default settings incldluing *vocabulary*, *embedding size* etc.

3.2 Siamese ALBERT

The central challenge of retrieving a proper answer lies in the complex and versatile semantic relations observed between questions and responses. Unlike several factoid QA scenarios (Rajpurkar et al., 2016; Kwiatkowski et al., 2019), linguistic similarities between such non-factoid QA might be non-indicative. Since ALBERT makes use of Transformer network (Vaswani et al., 2017), it benefits from multi-head attention mechanism, allowing the model to jointly attend to information from different representation subspaces at different positions.

[1]https://huggingface.co/models

3.2.1 Siamese Encoder

First, the ALBERT tokenizer split original words into smaller subwords and characters. After tokenization, two special tokens (i.e. [CLS] and [SEP]) are added to the tokenized sequence. Besides, the tokenizer also generates "Mask IDs", which is used to indicate which elements in the sequence are tokens and which are padding elements.

Second, Two ALBERT encoders generate contextualized representations of each tokenized element in an input question Q and a candidate answer A, which are denoted by $\text{ALBERT}_{\Theta}(Q)[X] \in \mathbb{R}^d$ and $\text{ALBERT}_{\Theta}(A)[X] \in \mathbb{R}^d$, where d is the dimension of word embedding. In order to produce a single vector of question embedding $\mathbf{h}_Q$ and answer embedding $\mathbf{h}_A$ from the input question and answer. We apply mean pooling to the representations of all tokens:

$$\mathbf{h}_Q = \textsc{Mean}(\{\text{ALBERT}_{\Theta}(Q)[X] | X \in Q\}), \quad (1)$$

$$\mathbf{h}_A = \textsc{Mean}(\{\text{ALBERT}_{\Theta}(A)[X] | X \in A\}). \quad (2)$$

3.2.2 Matching Layer

The scoring function F is factorized as an inner product between question embedding $\mathbf{h}_Q$ and answer embedding $\mathbf{h}_A$. This is similar to using feed forward networks to project queries and responses into a common space where the relevance is computed by cosine distance (Huang et al., 2013).

$$F(\mathbf{h}_Q, \mathbf{h}_A) = \frac{\mathbf{h}_Q}{||\mathbf{h}_Q||_2} \cdot \frac{\mathbf{h}_A}{||\mathbf{h}_A||_2}. \quad (3)$$

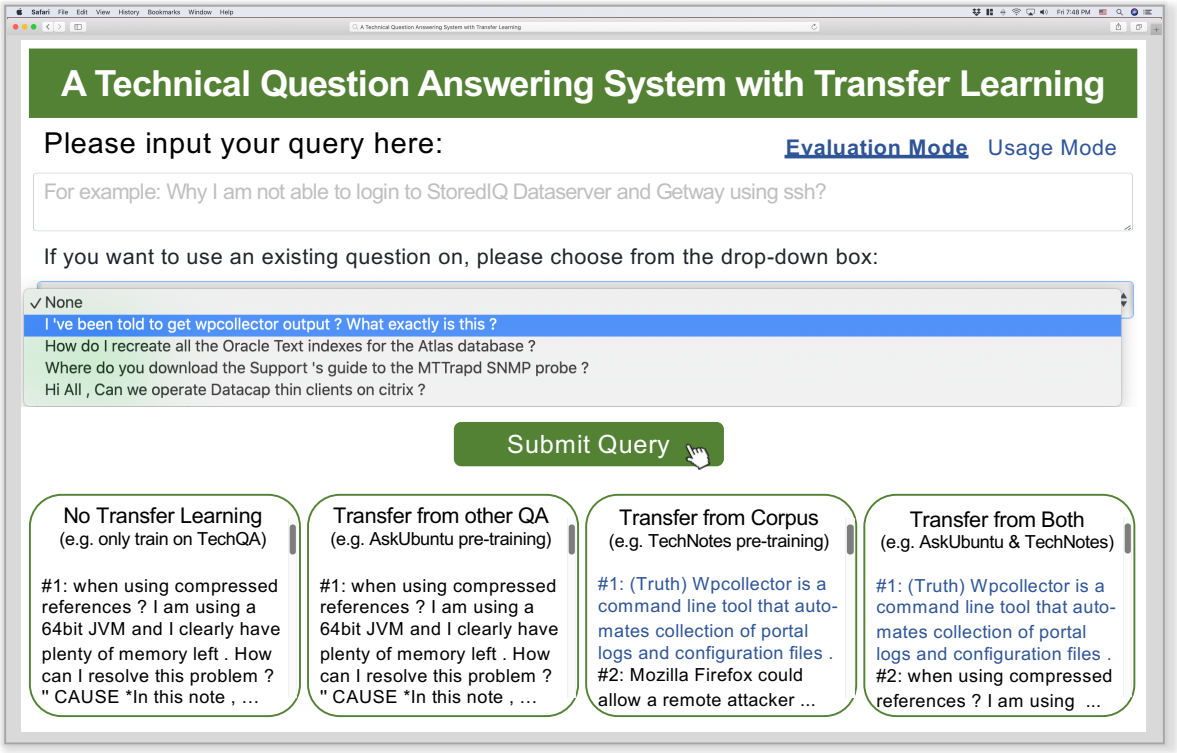

Figure 2: A screenshot of our system. User can either submit their query or choose a query from a pre-defined set. Our system will return top-3 recommended answers retrieved from the existing database.

3.2.3 Optimization

The goal of response selection is to model $P(A|Q)$, which is used to rank possible responses A given an input question Q. This probability distribution can be written as:

$$P(A|Q) = \frac{P(Q,A)}{\sum_k P(Q,A_k)} \tag{4}$$

For efficiency and simplicity, we adopt multiple negatives ranking loss in the training, which takes the responses of other examples in a training batch of stochastic gradient descent as negative responses (Henderson et al., 2017). Specifically, for a batch of size K, there will be K input questions $\mathcal{Q}_{batch} = (X_1, \cdots, X_K)$ and their corresponding responses $\mathcal{A}_{batch} = (A_1, \cdots, A_K)$. Every reply A_j is effectively treated as a negative candidate for Q_i if $i \neq j$. The $K-1$ negative examples for each Q are different at each pass through the data due to shuffling in stochastic gradient descent. So, the goal of training is to minimize the approximated mean negative log probability of the data. Formally,

$$\begin{aligned}\mathcal{L}(\mathcal{Q}_{batch}, \mathcal{A}_{batch}, \Theta) &= -\frac{1}{K}\sum_{i=1}^{K} \log P(A_i|Q_i) \\ &= -\frac{1}{K}\sum_{i=1}^{K}\left[F(\mathbf{h}_{Q_i}, \mathbf{h}_{A_i}) - \log \sum_{k=1}^{K} e^{F(\mathbf{h}_{Q_i}, \mathbf{h}_{A_k})}\right]\end{aligned} \tag{5}$$

3.3 Transfer: Three-step Fine Tuning

The transfer learning strategy used in our system consists of a three-step fine tuning process. First, the pre-trained ALBERT model is fine tuned with unstructured source technical corpus with masked language model (MLM) task, i.e., $\theta \rightarrow \theta^{+}$. Second, a siamese ALBERT takes fine tuned ALBERT and fine tunes with source technical QA, i.e., $\theta^{+} \rightarrow \theta^{++}$. Third, the siamese ALBERT further fine tunes with target QA, i.e., $\theta^{++} \rightarrow \theta^{+++}$.

3.3.1 Fine Tuning on Source Corpus

Though ALBERT has demonstrated state-of-the-art performances in many NLP tasks, directly applying ALBERT to technical domain tasks suffers from certain limitations. Since ALBERT is pre-trained on datasets only containing general domain corpora (e.g., Wikipedia, BookCorpus), many technical terminology are not well represented in the model. In order to make ALBERT equipped with technical domain knowledge, we first fine tune the pre-trained ALBERT on technical corpus (e.g., solution documents, user manuals and forum posts etc). We adopt the classic masked language model (MLM) pre-training task that learns to predict randomly masked tokens in the input sequence. MLM

Table 1: Performance on TechQA and StackUnix. Transferring knowledge from both source QA (e.g., AskUbuntu) and technical corpora (e.g., IBM Technotes) demonstrates the best performance.

Methods		Source QA	Source Corpus	TechQA MRR	TechQA R@1	TechQA R@5	TechQA R@10	StackUnix MRR	StackUnix R@1	StackUnix R@5	StackUnix R@10
SASE (LSTM)	M1	-	-	23.45	14.65	33.12	36.31	31.22	24.37	38.71	45.88
	M2	✔	-	26.19	17.20	35.03	42.68	34.03	27.24	40.01	45.16
	M3	-	✔	30.85	20.38	40.13	52.23	35.26	25.45	43.73	53.05
	M4	✔	✔	**36.31**	**25.48**	**48.41**	**56.05**	**38.69**	**29.30**	**46.24**	**55.20**
ALBERT	M1	-	-	36.52	25.00	50.63	56.25	40.55	31.89	50.18	56.99
	M2	✔	-	**45.11**	**34.38**	56.88	64.38	42.61	34.05	50.89	55.56
	M3	-	✔	41.61	29.38	55.00	63.13	46.27	35.12	58.06	67.03
	M4	✔	✔	44.13	31.88	**60.00**	**70.63**	**50.04**	**38.35**	**63.08**	**71.32**

is the primary pre-training task used in BERT and ALBERT (Devlin et al., 2019; Lan et al., 2020).

3.3.2 Fine Tuning on Source QA

Compared to evolved technical communities (e.g., StackOverflow, AskUbuntu), many emerging technical forums have limited size of existing QA pairs. To address the lack of knowledge caused by data shortage, we further fine tuned the ALBERT using source technical QA data to facilitate the ALBERT to better represent questions and responses in the latent space. For instance, since Ubuntu is a operating system built upon the Unix/Linux, many questions on AskUbuntu and StackUnix are related and share common technical knowledge.

3.3.3 Fine Tuning on Target QA

After fine tuning on unstructured technical corpora and source technical QA, a straightforward way to transfer these knowledge is to keep the same model architecture and directly initialize the weights of the target model with the weights of the models fine tuned on the source technical QA, and then we train on the target model with the target technical QA dataset (e.g., TECHQA, StackUnix).

4 Front End and User Interface

Our system consists of two different modes: evaluation mode and usage mode. First, the usage mode allows users to select an existing question from the drop-down box or ask a question themselves. Then, the system returns three retrieved responses with highest probabilities from the database to the user. Second, the evaluation mode retrieves responses through different transfer learning settings, which can be used to compare different model performance. Overall, the QA front end service was implemented in Python with Flask web framework. It is fully configurable and allows multiple candidate ranking services to be used at the same time.

Table 2: Comparison with existing retrieval QA system on number of parameters and average inference time. The siamese-ALBERT in our TransTQA system has the least parameters and fastest inference time.

Methods	#Parameters	Inference time
BERT-Rerank (Nogueira et al., 2019)	170M	62.67s
BERT-Serini (Yang et al., 2019)	170M	0.55s
Siamese-ALBERT (Ours TransTQA)	**10M**	**0.13s**

4.1 Usage Mode

To begin with, the system receives a question from the user via a text field input. The question is filtered by length: too long (>512) or empty questions are discarded. Then, the question is tokenized, padded and concatenated with special tokens. Given the tokenized query, the response retrieval reduces to encoding a new question in a encoding step to the question embedding, and then ranks all candidate answers based on the inner product with the query embedding. Since embeddings of candidate answers can be pre-computed, the usage mode is able to make quick response.

4.2 Evaluation Mode

In evaluation mode, when the system receives a query, the system retrieves responses from four different settings: (1) M1: no transfer learning; (2) M2: transfer knowledge from source corpus

Table 3: Case study of real user queries.

Case Study of Evaluation Mode
User Query: Hi All, Can we operate Datacap thin clients on citrixs ?
Ground Truth Answer: Remote users that access Datacap over a WAN can use Taskmaster Web-based thin clients , or FastDoc Capture operating in offline mode .
M1-Rank1: The upgrade must be initiated by using the attached script , which gives the non-root user (who originally installed this Jazz for Service Management instance) the correct permissions for the upgrade process .
M2-Rank1: WAS recently renamed WAS for Developers (WAS4D) to WAS ILAN . In response to this change , RAD 9.6 now packages WAS ILAN available in the RAD_WS_9.6_WASTE_9.0.zip file . It has the same capabilities and updates that RAD users expect .
M3-Rank1: You must install the 5.0.0.8 or later fix pack to upgrade to 6.0.0.0 or later . The 5.0.0.8 fix pack contains a required fix to allow the larger sized firmware image to fit .
M4-Rank1: Remote users that access Datacap over a WAN can use Taskmaster Web-based thin clients , or FastDoc Capture operating in offline mode .
Case Study of Usage Mode
User Query: Where do you download the Support 's guide to the MTTrapd SNMP probe ?
Ground Truth Answer: The attached Support 's guide to the SNMP probe provides details on how best to configure the probe , troubleshoot issues and how to use third party products to test the probes behaviour .
M4-Rank1: The attached Support 's guide to the SNMP probe provides details on how best to configure the probe , troubleshoot issues and how to use third party products to test the probes behaviour .
M4-Rank2: Supported methods are GUI and silent configuration . Also , silent configuration fully works both on UNIX and Windows platforms only at level 7.1.1.0.4 . Before this maintenance level there were issues and limitations with Managing Server and Transaction Tracking integration . Support Command Line configuration for J2EE DC will possibly be added in some future patch via Request For Enhancement .
M4-Rank3: This probe is written to support Nokia Network Functions Manager for Packet release 17.3 .

(e.g., Technotes); (3) M3: transfer knowledge from source QA (e.g., AskUbuntu); (4) M4: transfer learning from both (2) and (3). The retrieved responses are presented in four blocks corresponding with four different settings. In case the question was present in the dataset, because we know which one is the correct answer, we mark its text with a "Truth" tag and highlight in blue for a quicker performance assessment by the user.

5 System Evaluation

5.1 Experimental Settings

We conduct experiments on three technical datasets: TechQA (Castelli et al., 2020), StackUnix and an IBM internal dataset. Details about datasets are in Appendix **??**. We compare retrieval performance with SASE (Lin et al., 2017), which is a LSTM-based method adopted in previous QA systems (Loginova and Neumann, 2018; Rücklé and Gurevych, 2017). Implementation details can be found in Appendix **??**. We also compare inference time and memory consumption with two BERT-based QA systems: BERT-Rerank (Nogueira et al., 2019) and BERT-Serini (Yang et al., 2019).

5.2 Experimental Results

Comparisons with traditional QA systems As shown in Table 1, our TransTQA outperforms traditional QA system built upon LSTM-based models.

Effectiveness of knowledge transfer As shown in Table 1, knowledge transfer from source corpus and source QA are crucial for our task. We observe that fine tuning on both source corpus (e.g., Technotes) and source QA (e.g., AskUbuntu) makes superior performance than only fine tuning on either source domain corpus or source QA.

Time and memory efficiency As shown in Table 2, the siamese ALBERT is much faster than BERT-Rerank and BERT-Serini during the inference time. This is because all the answer embeddings stored in the database are pre-computed. So, given an unseen query, we only need to rank the answer based on its inner product with the query embedding.

5.3 Case Study

We show two real use cases in Table 3. Evaluation Mode returns top ranked answer retrieved by different transfer learning settings (M1 to M4). Usage Mode returns top ranked answer retrieved by M4. M4-Rank1 gives the correct answer.

6 Conclusions

We developed TransTQA, which is a novel system that offers automatic response by retrieving proper answers based on correctly answered similar questions. TransTQA was built upon a siamese ALBERT network, which enables it to respond to questions quickly and accurately. Furthermore, TransTQA adopted transfer learning to improve its performance on multiple tech domain QA.

Acknowledgements

This work is supported by National Science Foundation IIS-1849816.

References

Zahra Abbasiyantaeb and Saeedeh Momtazi. 2020. Text-based question answering from information retrieval and deep neural network perspectives: A survey. *arXiv preprint arXiv:2002.06612*.

Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. 2019. Reqa: An evaluation for end-to-end answer retrieval models. In *Proceedings of Workshop on Machine Reading for Question Answering*.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Mike McCawley, et al. 2020. The techqa dataset. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *In Proceedings of 8th International Conference for Learning Representation (ICLR)*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2019. Bidirectional attentive memory networks for question answering over knowledge bases. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers*, pages 2913–2923.

Yang Deng, Ying Shen, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. 2018. Knowledge as a bridge: Improving cross-domain answer selection with external knowledge. In *Proceedings of the 27th international conference on computational linguistics*, pages 3295–3305.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yunhsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. 2016. Little is much: Bridging cross-platform behaviors through overlapped crowds. In *AAAI*, pages 13–19. Citeseer.

Sosuke Kato, Riku Togashi, Hideyuki Maeda, Sumio Fujita, and Tetsuya Sakai. 2017. Lstm vs. bm25 for open-domain qa: A hands-on comparison of effectiveness and efficiency. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1309–1312.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *International Conference for Learning Representation (ICLR)*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *In Proceedings of 5th International Conference for Learning Representation (ICLR)*.

Ekaterina Loginova and Günter Neumann. 2018. An interactive web-interface for visualizing the inner workings of the question answering lstm. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 30–35.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Universal text representation from bert: An empirical study. *arXiv preprint arXiv:1910.07973*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 510–517.

Rodrigo Nogueira, , and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Andreas Rücklé and Iryna Gurevych. 2017. End-to-end non-factoid question answering with an interactive visualization of neural attention weights. In *Proceedings of ACL 2017, System Demonstrations*.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18.

Minjoon Seo, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Phrase-indexed question answering: A new challenge for scalable document comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Ivan Srba and Maria Bielikova. 2016. A comprehensive survey and classification of approaches for community question answering. *ACM Transactions on the Web (TWEB)*, 10(3):1–63.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473.

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

Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. 2019. Adversarial domain adaptation for machine reading comprehension. In *Proceedings of Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Lingfei Wu, Ian EH Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. 2018. Word mover's embedding: From word2vec to document embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2013–2018.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying bert to document retrieval with birch. In *Proceedings of Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*.

Wenhao Yu, Lingfei Wu, Qingkai Zeng, Yu Deng, Shu Tao, and Meng Jiang. 2020. Crossing variational autoencoders for answer retrieval. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.