DevRev

Mid-term Report

Expert Answers in a Flash: Improving Domain-Specific QA

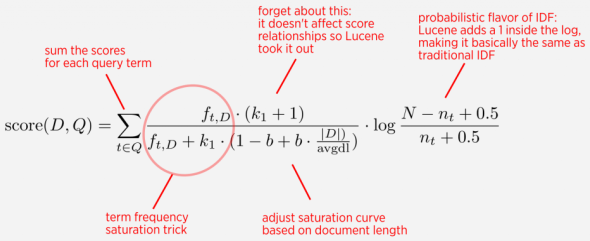
TeamID ***23***

The report summarizes various existing methods and related experiments on the given data. Also, results on the different reader and retriever architectures, answerability, and domain adaptation are reported. Note that all the experiments are done on **Colab CPU** on a train-test split with independent themes.

Architectures

Retriever

BM25



BM25 is a probabilistic model used in NLP to score the relevance of documents to a given query. It is based on TF-IDF and takes into account the frequency of a term in the document and the collection of documents. While SOTA models generally outperform BM25 in accuracy due to BM25's simplicity, we plan to use **BM25 under time constraints and follow it with a SOTA retriever.** Our findings are summarized in the table below.

Top k	Accuracy*	Average time per query (s)	Memory Usage
1	78.03%	0.30	4.50GB RAM
2	86.78%		
3	90.12%		
10	96.04%		

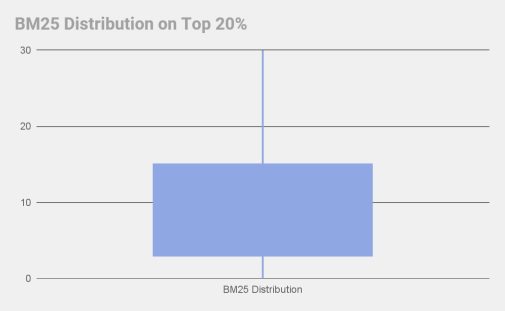
*Accuracy here means the presence of the correct document in the ranking produced by BM25.

Cross Encoder

Cross-encoder, often used as a re-ranker, takes as input a query and document and outputs a score between 0 and 1, **denoting the semantic similarity** between those two. It is not generally used as a primary ranker because the model has to compute a score for each pair of queries and document. It is trained by fine-tuning a base like BERT with its classifier component. Both the query and document are passed with a [SEP] token in between to the BERT encoder simultaneously, followed by a classifier head that calculates the similarity.

We experimented on miniLM and TinyBERT cross-encoder, which are pre-trained on the SQUAD 2.0 dataset. We explore the cross-encoder as a retriever, applying it to all documents in the theme and also as a ranker, where we **apply it to the top 10 paragraphs retrieved by BM25**. The table below summarizes the accuracy of the **presence of the correct paragraph among the top k candidate paragraphs.**

Model	Top k	Accuracy	Time per query (s)
TinyBERT cross-encoder applied on all paragraphs in the theme	1	85.02%	0.05
TinyBERT cross-encoder applied on top 10 paragraphs based on BM25	1	86%	0.02
MiniLM cross-encoder applied on top 10 paragraphs based on BM25	1 (top 1 of MiniLM)	88.71%	0.76
	2 (top 1 of BM25 and MiniLM each)	90.92%	0.76
MiniLM, and TinyBERT cross-encoder applied on top 10 paragraphs based on BM25 separately	3 (top 1 of BM25, MiniLM and TinyBERT each)	92.51%	0.82



We need to maximize the chance that the correct paragraph is among the top k decided by the BM25. Instead of setting k as a fixed value like 10, we hypothesized that it might be better to **choose k based on quartiles**. This is because if we were to take k = 3, even the 4th paragraph could be the correct paragraph if its score is similar to the 3rd one. Hence, a variable amount is taken based on the distribution of scores of the BM25 retriever. If we pass the top 20% to the tinyBERT ranker, we see that the top 1 prediction improves by nearly 1%. The box plot shows the distribution of the number of paragraphs present in the top 20% decided by the BM25 scores.

Dense Passage Retrieval (DPR)

Dense Passage Retrieval [1] (Karpukhin et al. 2020) is used for passage retrieval, and it theoretically outperforms TF-IDF and BM25 by 9-19% accuracy as it is **able to capture the semantics**. It uses two separate BERT encoders, out of which one encodes passages of text into a vector while the other model similarly encodes the question. During training, the model weights will be optimized to maximize the dot product between two encoding vectors of the respective model outputs. By training the two models to give a similar vector output, we are **training the context encoder and question encoder to output very similar vectors for**

related question-context pairs. The paragraphs are then ranked based on the dot-product value after passing through the encoder. The following table summarizes the top k accuracy of a pre-trained DPR model. As it can be seen, in practical purposes, the previous models outperform DPR, and hence it is not further used.

Top k	Accuracy*
1	51.46%
2	66.01%
3	73.48%

**Accuracy here means the presence of the correct document in the ranking produced by DPR.*

Bi-encoder

Bi-encoder computes vector representations for query and paragraph. Then, **cosine similarity** between these vector representations is used to retrieve the most similar candidate paragraph. An advantage over cross encoder is the **ability to precompute** the document and query embeddings and hence is considerably faster. We found that the top 1 accuracy in retrieval is 56.77% and took 99.97 ms per query on average.

Reader

Transformer for Question Answering

Transformer architecture [2] (Vaswani et al. 2017) for extractive Question Answering (QA) tasks involves **predicting the starting and the ending token probabilities** using an encoder-decoder architecture. The tokens between the two maximum probability start and end tokens are taken as the extracted answer. The following table summarizes our experiment on pre-trained readers

Model	Accuracy (Exact Match)	Time (per query)	Memory	F1 score
roberta-base-squad2	83.27%	2020 ms	496 MB	62.33
roberta-large-squad2	89.98%	6500 ms	1420 MB	70.32
tinyroberta-squad2	79.26%	630 ms	326 MB	66.73
minilm-uncased-squad2	78.85%	305 ms	134 MB	64.85

distilbert-base	51.67%	474 ms	261 MB	45.50

Here, F1 scores are calculated based on the given sample evaluation code. As it can be seen, even the lightest of the readers can't perform within 200ms, which calls for **easing the given constraint.**

Retro Reader

We tried Retro Reader [3] (Zhang et al. 2020), one of the current state-of-the-art models for reading comprehension. It performs **reading in 3 stages**:

1. Sketchy reading that briefly touches the relationship of passage and question using an external verifier (E-FV)
2. Intensive reading that verifies the answer and gives the final prediction using an internal front verifier (I-FV)
3. Finally, there is a Rear Verification(RV), a combination of predicted probabilities of E-FV and I-FV, which is taken as the aggregated verification for the final answer.

Model	Time (per query)	Accuracy (Exact Match)	Memory	F1 Score
Retro Reader	13.96s	90.56%	3.86 GB	87.76%

We tested Retro Reader with the base as electra_large on approx 1000 examples (due to time constraint). The model will give similar results for inference: **the model has high accuracy but high latency, which limits its use.**

Knowledge Distillation

As many of the SOTA NLP models have a very high usage of storage and are computationally very expensive, we tried using **distillation in order to produce good results on much lighter models.** The process involves a smaller student model trying to generalize and imitate a huge teacher model like BERT large. The smaller model learns from the output of the teacher model on the dataset and the ground truth values of the dataset. We did not use only the target class's label to train the student but the whole softmax layer of the teacher to train it.

Keeping the advantages of distillation in mind, we have **distilled BERT-large to TinyBERT** with prediction and intermediate layer distillation. We leveraged **data augmentation** (explained below) to increase the number of training samples by a factor of 2. However, training the intermediate layer takes several days as the data has been augmented; hence the results are yet to be evaluated.

Answerability

The problem of answerability given a paragraph and a context (task 1) is addressed in various ways in the existing literature. For example, a retro-reader, as outlined earlier, determines the answerability with the help of an internal front-verifier.

Many pre-trained reader models have output the confidence score of the prediction, which is the sum of the logits of the start token and end token produced by the final layer of the reader converted into a pseudo-probability. A common practice is to take **0.5 as the threshold** and consider answers with a confidence of less than 0.5 as unanswerable. Our experiments reveal that applying the threshold on a pre-trained tinyroberta model predicts the answerability 95.8% of the time.

We propose a novel method in which we use the **confidence score of both the reader and the retriever to determine the answerability** of a question. The intuition is that the retriever scores must also be high if the reader's confidence is high. We thus fit a perceptron classifier on the reader, ranker, and top 10 BM25 scores to classify whether the paragraph can be answered or not. Upon experimentation, the hypothesis is validated, as can be observed in the table below:

Method	**Data**	**Accuracy**	**F1**
0.5 Threshold	Reader Score	95.80%	96.34%
Perceptron Classifier	Top 10 Retriever Score	69.57%	79.41%
Perceptron Classifier	Top 10 Retriever + Reader Score	97.46%	98.17%
Perceptron Classifier	Top 10 Retriever + Ranker + Reader Score	97.61%	98.53%

Here Retriever is BM25, Reader is TinyRoBERTa and Ranker is miniLM cross-encoder.

Domain Adaptation

Retriever

Generative Pseudo-labeling (GPL)

Generative pseudo-labeling [4] (Wang et al. 2021) is a method for **adapting an NLP model to a new domain using a small amount of labeled data and a large amount of unlabeled data.** A pre-trained T5 model is used to generate queries from the unlabeled data, and a pre-trained retriever is used to retrieve similar paragraphs (including negative examples). A cross-encoder is then used to score each pair and fine-tune the retriever.

In the original paper, GPL achieved state-of-the-art results in domain-specific applications. However, when adapting the model to our domain using the 'msmarco-distilbert-base-tas-b' retriever, fine-tuning with GPL unexpectedly resulted in a decrease in accuracy from **0.9** to **0.82**, possibly due to overfitting as the MS-MACRO dataset, which was used to train the model, could have already be trained on the required domains.

Reader

Fine Tuning

Finetuning the reader architecture on the given dataset was hypothesized to improve the exact match accuracy of the model. So two experiments were conducted to test the hypothesis corresponding to round 1 and round 2 tasks.

First, we finetune a pre-trained reader on a **train-test split without any common themes** to test adaptability in the round 1 setting. The result is outlined in the following table:

Model	**Pretrained Dataset**	**Accuracy Before Training**	**Accuracy After Training**
MetaQA	SQUAD, Duorc, Race	40.70%	44.14%
TinyRoBERTa	SQUAD	79.26%	28.37%

We notice that **any reader pre-trained exclusively on the SQUAD dataset had a sharp decrease in the accuracy**. This can be explained as the given dataset is quite similar to the SQUAD dataset, and hence training on a subset of the pre-trained dataset will result in

overfitting. However, we observe a steady increase in the accuracy if we finetune a model such as MetaQA that is also pre-trained on other QA datasets, as indicated by the table above.

Secondly, we finetune a pre-trained miniLM model on a train-test split with **perfectly overlapping themes** to gauge the performance in theme-adaptability (round 2 task). However, training the model resulted in a **slight decrease in accuracy** with the number of epochs. The same is observed in the following table:

Epochs	**Accuracy**
Pretrained	78.87%
Epoch 2	77.93%
Epoch 3	77.61%

Task Transfer

Task transfer [5] (Pan et al. 2022) is a domain adaptation technique in which a Transformer for QA like RoBERTa is trained for a task like a language modeling or named entity recognition on a corpus of a specific domain. Then the model is **fine-tuned for generic question-answering tasks** on the given dataset. An increase in the F1 score for question-answering on that specific domain was reported by the authors.

Using Theme Embeddings

An additional way to incorporate domain knowledge is by **concatenating the theme name as an extra representation while training.** We will be making use of a pre-trained RoBERTa model and will be finetuning it over the given training data. Doing so, we observed a **slight increase** in the test accuracy from 78.85% to 79.13% while predicting with miniLM.

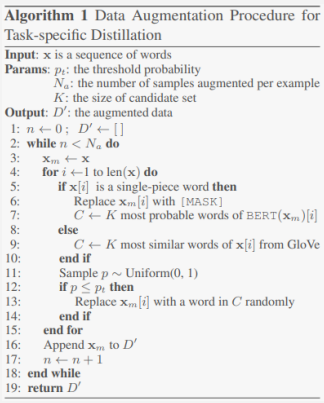
Data Augmentation

The following augmentations were done on the given dataset for fine-tuning the models mentioned earlier.

Original Data Augmentation

We performed data augmentation on the dataset to increase training data so that we could finetune the TinyBERT model for our dataset in order to accelerate inference by using the

knowledge distillation model. For augmenting the data, we combine a pre-trained language model BERT and GloVe word embeddings to do **word-level replacement for data augmentation**. Specifically, we used the transformer model and the tokenizer from the language model *bert-base-uncased* to predict word replacements for single-piece words and use the word embeddings to retrieve the most similar words as word replacements for multiple-pieces words. We set pt = 0.4, Na = 20, and K = 16 for the augmentation hyperparameters given in Algorithm 1.



Algorithm 1 Data Augmentation Procedure for Task-specific Distillation

Input: $\mathbf{x}$ is a sequence of words
Params: p_t: the threshold probability
N_a: the number of samples augmented per example
K: the size of candidate set
Output: D': the augmented data

$n \leftarrow 0$; $D' \leftarrow [\,]$
while $n < N_a$ **do**
  $\mathbf{x}_m \leftarrow \mathbf{x}$
  for $i \leftarrow 1$ to len($\mathbf{x}$) **do**
    if $\mathbf{x}[i]$ is a single-piece word **then**
      Replace $\mathbf{x}_m[i]$ with `[MASK]`
      $C \leftarrow K$ most probable words of $\texttt{BERT}(\mathbf{x}_m)[i]$
    else
      $C \leftarrow K$ most similar words of $\mathbf{x}[i]$ from GloVe
    end if
    Sample $p \sim$ Uniform(0, 1)
    if $p \leq p_t$ **then**
      Replace $\mathbf{x}_m[i]$ with a word in C randomly
    end if
  end for
  Append $\mathbf{x}_m$ to D'
  $n \leftarrow n + 1$
end while
return D'

Adversarial Data Augmentation

Adversarial data augmentation [6] (Max Bartolo et al. 2022) uses **generative models like BART to produce challenging questions based on candidate answer phrases**, and this question-answer pair is used for training. In practice, adversarial data augmentation has proven to be successful at improving the model's performance and generalizability.

Other Augmentations

To increase the negative examples, we pair a question with a paragraph that doesn't contain the answer to that question. A **hard negative** (negative examples that are hard for the model) could be identified if the reader gives a high confidence score for the selected negative paragraph.

Future Work

Proposed Pipeline

Based on the experiments above, we plan to use BM25 as the **first stage retriever** due to its low latency and the fact that there is a probability of 96.04% that the correct paragraph lies in the top 10 paragraphs fetched by BM25. As explained earlier, we plan to take the **top 20%** of the documents for the second stage of retrieval. Then, we proposed to use one or two cross-encoder (depending on the time constraint) to **re-rank** the documents as we have **narrowed down the search space.**

To improve the latency, if the top predicted paragraph is the same for the two rankings, we pass it along to the reader stage. As it is **verified by two rankers**, there is a minimal chance it is not the correct paragraph. If the top predicted paragraph is not the same, we must decide the

top 2 among the **normalized scores of the two rankings**. This could result in either picking the top 2 of either ranker or the top 1 of each ranker.

We then evaluate the answer on the paragraph(s) using a distilled (depending on the time constraint) and finetuned reader, which will produce the answer text with a confidence score. As described earlier, a **perceptron classifier is used to decide the answerability**, and the final answer is outputted appropriately.

The pipeline architecture is subject to change upon changes in the time constraint.

Code

The code for the experiments conducted can be found here.

References

[1] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. arXiv. https://doi.org/10.48550/arXiv.2004.04906

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *arXiv*. https://doi.org/10.48550/arXiv.1706.03762

[3] Zhang, Z., Yang, J., & Zhao, H. (2020). Retrospective Reader for Machine Reading Comprehension. *arXiv*. https://doi.org/10.48550/arXiv.2001.09694

[4] Wang, K., Thakur, N., Reimers, N., & Gurevych, I. (2021). GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. *arXiv*. https://doi.org/10.48550/arXiv.2112.07577

[5] Pan, X., Sheng, A., Shimshoni, D., Singhal, A., Rosenthal, S., & Sil, A. (2022). Task Transfer and Domain Adaptation for Zero-Shot Question Answering. arXiv. https://doi.org/10.48550/arXiv.2206.06705

[6] Bartolo, M., Thrush, T., Jia, R., Riedel, S., Stenetorp, P., & Kiela, D. (2021). Improving Question Answering Model Robustness with Synthetic Adversarial Data Generation. *arXiv*. https://doi.org/10.18653/v1/2021.emnlp-main.696