Inter-IIT Tech Meet-11.0 DevRev

Team No 31

January, 2023

1 Abstract

Passage Retrieval and Question Answering are essential tasks in developing models capable of Answering Natural Language questions over a wide range of topics. Question Answering finds its applications in information retrieval, customer service, and knowledge management. In this work, we implemented a two-stage framework that combines a context retriever and a reader component. The retriever component is responsible for retrieving a context from an extensive knowledge base of contexts relevant to the input question. The reader component then takes the retrieved contexts and thoroughly examines the contexts to extract the most appropriate answer. We used synthetic data to overcome data scarcity and bias in the model. Generating synthetic training data and using Question Similarity to match the Questions with already answered questions lead to improved performance and robustness. The use of a retriever-reader architecture has shown promising results. It is widely used in various Question Answering Systems, so we chose this approach. The later part of the work will provide an overview of this architecture and discuss the different strategies and techniques like Synthetic Data Generation and Question Similarity Matching we used to decrease the inference time and to increase the performance of the retriever and reader components.

2 Implementation

2.1 Synthetic Data Generation

Model Details

For generating synthetic data, we have used a answer aware generation model and a model to generate answers. The answer generation model generates the answers as spans. The answer aware question generation model takes in the given answer spans and generate questions corresponding to it. We first discuss the answer generation method that was used and discuss the answer aware question generation model.

For generating the answer as spans multiple methods exist such as Named Entity Recognition, Noun phrase extraction etc. But those models limit the diversity in the kind of answers that are extracted. And also it doesn't use any information from the training set to get a sense of what kind of questions are important. Thus we train a *T5* [12] model to extract the spans from the given passages. We iterate through each of the sentences and from each of the sentences we extract a span which will be considered to be our answer.

Using the spans that were generated by the previous model, we generate a corresponding question which has the span as its answer. We use another T5 model which is fine tuned on this task. While giving the inputs we highlight the answer spans with a <hl> token [1]. If we had given the answer as separated token at the end or beginning, it would have been difficult for the model to infer about the span since the same answer tokens can repeat at different places in the sentence. Beam search was employed while generating the questions.

We also tried to train the models which predict both question and answers in a single pass [15]. The decoder models are trained to predict the questions, answers separted by special tokens. The models are easier to train and take a bit less space. But the quality of the questions generated were not very good. Since we use the synthetic data even in the inference time directly, we decided that the quality data is more important than the speed of the model.

2.2 Passage Retrieval Task

2.2.1 Dense Passage Retrieval

Training Methodology

We used facebook Dense Passage Retrieval (DPR) [8] for passage retrieval. We used the Facebook implementation of DPR [6]. We trained it on the initial 361 themes given to us. Internally DPR uses hugging face tokenizers. By default, the DPR model uses hugging face BERT-based [2] as the encoder. To select the paragraph with the same context as the question, we encoded the theme as a new token <theme>. We appended it to both question and the corresponding paragraph, which was used as an example of a positive context. We picked a random paragraph of a different theme with its token in front of it as an example of a negative context while Training the DPR. We split the data of each theme into train and dev in the ratio of 9:1, respectively. We used dev as validation data to check whether the model was overfitting. The model gave a top-1 accuracy of 0.748 as the validation accuracy for passage retrieval.

Fine Tuning Methodology

We finetuned the model on the 30 themes given to us on the final day and made it ready to run on the test data. Finetuning increased the model's performance, and the top-1 accuracy reported for the same test dataset after finetuning the above model was around 0.804. But we didn't implement the DPR model for passage retrieval because the inference time was high as we were sending the inference questions one by one. The DPR implementation of Facebook focused on optimizing the inference time when a list of inference queries is sent to get a list of passages inferred based on the questions. The inference time is significantly high when you send questions one by one as the model builds its parameters in the backend for each question again. Because of this issue, we shifted to the hugging face MultiQA-distilbert-cosv1 [4] model.

2.2.2 Final Implemented pipeline

We loaded the MultiOA-distilbert-cosv1 model from the hugging face Senctence transformers. We fine-tuned the pre-trained model using the initial 361 themes given to us. But the accuracy of the model didn't improve due to fine-tuning. The passage retrieval model gave an accuracy of 0.84 before fine-tuning, and even after fine-tuning, it gave roughly the same accuracy of 0.84. Training on the provided data could have been a better option, given the time and computational cost required, if the accuracy improved. But we didn't find any improvement in the accuracy, So we decided not to fine-tune it any further, and we used the pre-trained model directly.

2.3 Question Answering Task

2.3.1 Training Methodology

The model we used for Question Answering Task is "*roberta-base-squad2[5]*". This is a pre-trained model available on HuggingFace. This is the roberta-base model[13], fine-tuned using the SQuAD2.0 dataset[13]. It's been trained on question-answer pairs, including unanswerable questions, for the task of Question Answering. The model was directly tested on the dataset of 300 paragraphs and the corresponding question answers that was previously sent to train the model. The model had an exact match of 0.63 and an f1 score of 0.69.

2.3.2 Fine Tuning Methodology

The model we took was already fine-tuned on SQuAD2.0 dataset. To further improve its performance, we trained the model on the 300 paragraphs and corresponding question answers. On executing the model after training, we noticed that there was no improvement in the performance. We did extensive research to find a reason for the same and realized that the model overfits on the training data. So, there is an improvement in train data score but the model shows no improvement on validation/test data. Hence, we preferred not to train the model on the data provided to save computation time and cost. On receiving the 30 new themes, we trained the model on the new data but the there was no improvement in the scores. So, we proceeded without training on the data provided.

2.3.3 Final Implemented pipeline

After trying various pre-trained models and fine tuning them, we finally came to the conclusion that further training the models on training data is providing no benefits and is only leading to an increase in computation cost and time. Furthermore, roberta-base-squad2 has been selected over other question answering models keeping in mind the constraints that we have to follow and the best model.

2.4 Using already Answered Questions

For considering the previously answered questions in the data, we use a `MiniLM`[17] neural network like architecture, which is a multi layer transformer with self-attention. In the following sections we shall cover more detailed view of the same.

Our pre-trained model is a 6-Layer Transformer which contains 768 hidden size, and 6 attention heads which is previously trained on 1B sentence pairs dataset. This pre-trained model was intended to be used as a sentence and short paragraph encoder. Given an input text, it ouptuts a vector which captures the semantic information. The sentence vector can then be used for information retrieval, sentence similarity tasks or in our case create embeddings for paragraphs which will also be used to check similarity with the testing question. This pre-trained model is thus fit for the job of predicting which sentence was our input sentence paired with in the given random sample of several other sentences. This pre-trained model is used with the Hugging Face API[3]. We have used this pre-trained model on Hugging Face itself, to further fine tune it as per our problem's requirements.

2.4.1 Fine Tuning and Storage

As mentioned above we have taken the pre-trained model from the Hugging face API, which is performs great at finding similarity between sentences. We further fine tune this pre-trained model on Quora Question Pairs Dataset[11], which has pairs of question as input features and labels as binary classes (0 or 1) representing if the pair of questions are almost same(i.e. have same answers) or not. This fine tuning improves the performance of the model on finding questions which have same answer.

We effectively convert the question text into tensors, which we then use to find the similarity among the questions already answered and the new test question.

The key point here is that as we keep answering the unique new unanswered questions we keep on augmenting our stored tensor(the embeddings of training questions) with these new questions' embeddings. Along with this data, we keep track of the paragraph id, of the paragraph which answers the respective questions. Then using the cosine similarity between these tensors, we can find the most similar questions, this similarity captures the notion of questions having same answer. If the similarity value is more than the learnt threshold by the model, we directly answer them using the corresponding paragraph id's paragraph, which is mapped to the similar question in stored data. This stored data is available on the drive and we use the embeddings file from there only. Also this file uses very less storage(nearly 5 MB, for train data) and thus is feasible for us to use.

The main advantage that it gives us is that the overall inference time decreases significantly. This is because in stead of using the paragraph retrival model and then Quesiton Answering model, which take up a lot of time, we run this MiniLM model which is computationally very light and runs one forward pass in very less time.

2.4.2 Results

Following are the results of before ans after the finetuning :-

Metric	Before Fine Tuning	After Fine Tuning
Accuracy	0.787	0.835
F1 Score	0.743	0.796
Precision	0.631	0.720
Recall	0.902	0.891

Table 1: Metrics on Finding Questions with same Answers

Also the model take less than 3.443 seconds to evaluate/infer for 300 question pairs. This means that per question the inference time is roughly around 11.5 milliseconds on CPU.

3 Final Inference Notebook

In the final inference notebook we the pipeline is visualized in 1

For each of theme we will pre-process the passages to get the encoding from the models. We also store the encoding of the questions from the synthetic data. We also store the questions and answers from the synthetic data and the training data that we had generated. As the new questions come we encode the questions using the question similarity model that we have trained and find the similarity between the questions. If the similarity is greater than the threshold, we give the answers from the already trained model.

If the question doesn't match any previously generated questions, it is passed to the retriever model which generates the embeddings for the question and compares it with other all the embeddings of the passages by taking the cos similarity between them. The passage which has the highest matching is taken and the answer generation model is ran on it. The generated answers are again added to the dictionary of already answered questions so that if the number of testing questions is huge with little variance, the pipeline gives the answer very fast.

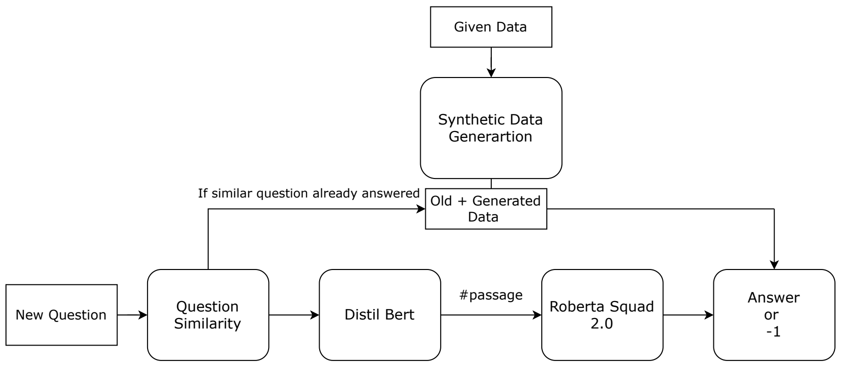

Figure 1: Final pipeline of the inference notebook

4 Results and Time Analysis

Model	F1 score	Average per Query Time in CPU
Dense Passage Retrieval (DPR)	0.884	0.002 sec/query
MultiOA-distilbert-cosv1	0.842	0.115 sec/query
MultiQA-distilbert-dotv1	0.803	0.115 sec/query

As we can see, even though the DPR has a bit better accuracy, its inference time is very high. Thus we had a trade-off between accuracy and inference time, and we chose MultiQA-distilbert-cosv1 instead of DPR. We can see MultiQA-distilbert-dotv1 and MultiQA-distilbert-dotv1 take almost the same inference time, but MultiOA-distilbert-cosv1 has a bit more accuracy hence we preferred MultiOA-distilbert-cosv1.

Retriever	Reader	final_para_score	final_qa_score
MPNet [16]	Dynamic-TinyBERT	0.53	0.46
MPNet	ALBERT	0.24	0.22
DistilRoBERTa [14]	Dynamic-TinyBERT	0.51	0.45
DistilRoBERTa	ALBERT	0.24	0.21
DistilBERT [14]	**Dynamic-TinyBERT**	**0.56**	**0.49**
DistilBERT	ALBERT	0.27	0.24

Among the models we tried till Midterm submission,although the models similar to ALBERT architecture are very promising, their *inference time* is quite high resulting in the extremely low values of *final_para_score* and *final_qa_score*.The Dynamic-TinyBERT outperforms all other models due to its extremely low *inference time* compared to other models and appreciable inference score.

The Average Inference Time analysis for various reader model are as follows:

Model Name	avg_inf_time(in sec)
RoBERTa	0.50
ALBERT[10]	0.76
ColBERT[9]	0.68
Dynamic-TinyBERT[7]	**0.38**

After the introduction of the question similarity model the time of inference decreased significantly. For the model with RoBERTa and DistilBERT-cos-v1 the total time in the pipeline was about **1.1 seconds** now came down to **0.30 seconds**.

5 Future Work

In the synthetic data generation models trying to optimize the size of an end-to-end model and improving its accuracy is a interesting problem. Also we have to look at improving the size of passage retrieval models. While generating the questions we have not looked at generating questions which are relevant to the given passage but do not have the answer in the given paragraphs. We are currently saving all the questions from the previous data sets. But we can develop a caching mechanism which keeps only the most important questions. This would help in decreasing the space requirements when the size of the dataset is large.

6 Conclusion

In conclusion, the field of Question Answering has seen significant progress in recent years, with the use of pre-trained language models, attention mechanisms, and neural networks contributing to the accuracy and efficiency of the Question Answering systems. The use of synthetic data has also emerged as a promising approach to overcoming data scarcity in QA systems. The report describes how we approached solving the Question Answering task given to us. The report shows the benefits of using techniques like Question Similarity Matching and Synthetic Data Generation to increase the performance of our QA systems. The implementation of question similarity matching has also improved the efficiency of the retrieval process, reducing the amount of irrelevant information that needs to be processed. Despite these improvements, queries are still to be addressed in question answering, such as ensuring the reliability and accuracy of the answers and dealing with ambiguity and vagueness in natural language questions. The continued progress in the field and the implementation of advanced techniques will likely play a crucial role in shaping the future of question-answering.

References

[1] Ying-Hong Chan and Yao-Chung Fan. A recurrent bert-based model for question generation. In *Proceedings of the 2nd workshop on machine reading for question answering*, pages 154–162, 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Hugging Face. all-minilm-l6-v2. `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`, 2022.

[4] Hugging Face. Multiqa-distilbert-cosv1 model. `https://huggingface.co/sentence-transformers/multi-qa-distilbert-cos-v1?doi=true`, 2022.

[5] Hugging Face. roberta-base-squad2. `https://huggingface.co/deepset/roberta-base-squad2`, 2022.

[6] Facebook. Dpr implementation. `https://github.com/facebookresearch/DPR`, 2022.

[7] Shira Guskin, Moshe Wasserblat, Ke Ding, and Gyuwan Kim. Dynamic-tinybert: Boost tinybert's inference efficiency by dynamic sequence length, 2021.

[8] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[9] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020.

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

[11] Kaggle Quora. Quora question pair dataset. `https://www.kaggle.com/competitions/quora-question-pairs/data`, 2017.

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

[13] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[14] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[15] Siamak Shakeri, Cicero Nogueira dos Santos, Henry Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. End-to-end synthetic data generation for domain adaptation of question answering systems, 2020.

[16] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding, 2020.

[17] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.