Expert Answers in a Flash: Improving Domain-Specific QA

Team ID: 49

***Abstract*—This study presents a natural language processing (NLP) based pipeline for question answering resource-constrained environments. Our pipeline takes a question and a set of paragraphs as input and uses two state-of-the-art NLP models, BM 250KAPI and Mini LM, to rank and classify the paragraphs based on their relevance to the question. The top-ranked paragraphs are then further processed by a finetuned Mini LM model for question answering, that outputs the start and end indices of the answer within the paragraph. The results of this pipeline demonstrate the effectiveness of using advanced NLP techniques for question answering in resource-constrained environments and highlight the potential for further refinement and optimization of the pipeline. This study contributes to the ongoing effort to develop and improve NLP-based question answering systems and has the potential to impact a wide range of applications.**

I. Implementation

We implement system for answering questions from a given paragraph that involves two main components: rankBM25 [1] for retrieving the top 10 relevant paragraphs, and a deep learning pipeline using MiniLM [2] for predicting the answer index from the relevant paragraphs. The rankBM25 method uses the BM25 ranking function to score and ranks the paragraphs based on their relevance to the input question. The top 10 paragraphs are then passed to the deep learning pipeline, which concatenates the paragraph and the question into a single block of text and processes it using the pre-trained MiniLM model. The MiniLM model is fine-tuned for binary classification (task 1) to determine whether the paragraph contains the information needed to answer the question, and for sequence labeling (task 2) to identify the starting indices of the answer within the paragraph. The pipeline output is the final answer to the question, that is obtained by converting the predicted index back into a text span from the original paragraph.

A. Paragraph retrieval task

1) Synthetic Data generation : We plan to expand the size of the dataset in the hope that the model will perform better when trained on more data. One method we aim to augment the data is by restructuring the data from active to passive or vice versa. We notice that there is a heavy intraclass-imbalance in the dataset – the overall distribution of classes in the dataset is balanced, but within each group of examples (in this case, paragraphs), there is an imbalance between the two classes (Paragraphs where the answer is located and not located). For example, within the paragraph corresponding to the theme Beyonce, we found zero questions in the provided dataset where the answer was not present in the paragraph. We fix this by formulating a method where we take the answer-start, removes the sentence corresponding to that index (and therefore in effect, it removes the answer), and updates the Answer possible parameter to False. We applies this rule across all themes where this imbalance is observed. This expands our dataset significantly.

2) Training methodology : We initially began with basing word embeddings such as Word2Vec combined with K-Nearest Neighbours We used the rank-bm api to get the top 10 probable paragraphs that might answer the given question. Rank-bmi is a collection of algorithms for "querying a set of documents and returning the ones most relevant to the query." We have used the BM250KAPI algorithm to retrieve the top 10 paragraphs. The number of paragraphs that we retrieved was chozen after analyzing the tradeoff between more paragraphs and time constraint.

In order to determine the most effective model for task 1, a comprehensive evaluation of several state-of-the-art models was conducted. These models included BERT [3], MiniLM, RoBERTa [4], ALBERT [5], and DistilBERT [6]. After carefully analyzing the results, it was determined that MiniLM exhibited the highest level of performance, and was therefore chosen as the final model for further testing and refinement. Refer to Table I for comprehensive experimentation results.

3) Run time improvements: Using rankBM25 improved the runtime efficiency of the system by reducing the number of paragraphs that needed to be processed by the deep learning pipeline using MiniLM. Instead of processing all the available paragraphs, rankBM25 was used to retrieve only the top 10 most relevant paragraphs based on their relevance to the input question. This reduced the number of input sequences that needed to be processed by MiniLM, which can be a computationally expensive task.

We considered various methods for improving the runtime efficiency of the system, including model pruning methods like quantization. However, we found that these methods resulted in a significant decrease in accuracy, so we decided to stick with the non-quantized model. Quantization involves reducing the precision of the model weights and activations, which can lead to faster processing times and lower memory requirements. However, this can also result in a loss of information and a decrease in accuracy. In our case, the decrease in accuracy was substantial, so we decided to use the non-quantized model in order to maintain a high level

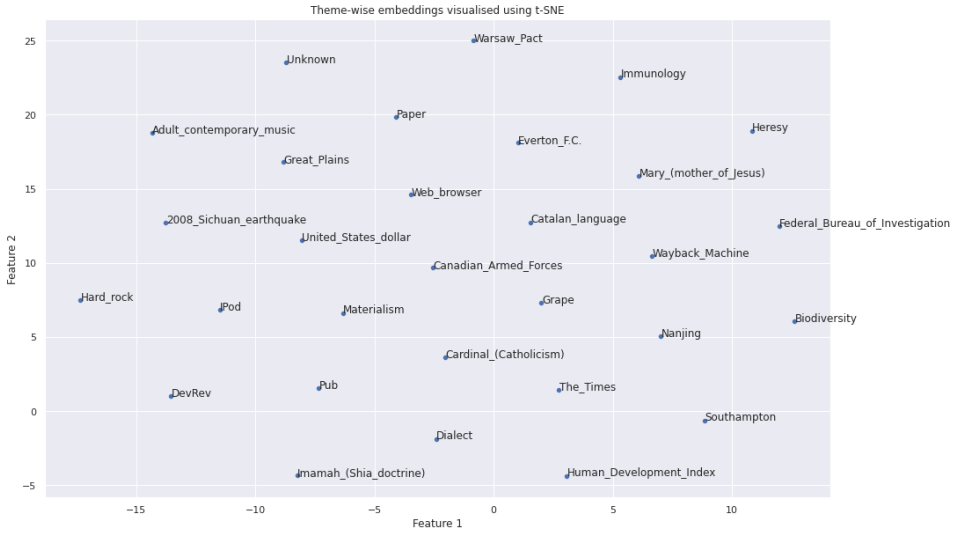

Fig. 1: t-SNE plot of theme-wise embeddings

Model	Training Accuracy	validation Accuracy	Test Accuracy	Parameters
BERT	93.20	83.34	83.35	110M
ALBERT	96.332	87.34	87.34	12M
RoBERTa	95.689	89.8	89.8	125M
DistilBERT	482.015	79.38	79.38	65M
MiniLM	93.116	89.76	89.77	33M

TABLE I: Model comparison

of accuracy. This trade-off between accuracy and runtime efficiency highlights the importance of careful consideration of these factors when developing a deep learning pipeline for a specific application.

B. Question answering task

1) Synthetic Data generation : We wrote a snippet that would jumble sentences in the paragraph and retrieve the index of the original answer, and reinjected it into the dataset. This significantly increased the number of datapoints available, specially for the task of extracting the answer token from a given paragraph. Another direction in which we proceeded was to use Google's PEGASUS [7] model to paraphrase these texts, and identified instances where the answer was preserved and where it was not, and then those instances were accordingly fed back into the dataset, grouped into their corresponding categories. Though this method was time intensive and we dropped it from the final approach.

2) Training methodology: From our inference of Task1 we did our experimentation on two specific models, DistilBERT and MiniLM. These were evaluated for the Question-Answering task (task 2). MiniLM has half the number of parameters, which directed us to use it for the Question Answering task too. Furthermore, the score achieved on MiniLM and DistilBERT were comparable.

3) Fine tuning methodology: In this section, we will be examining two different approaches to solve a particular problem. The first approach involves creating thirty separate models, each specifically designed for a unique theme. The second approach is to create a single model that can handle all thirty themes. After conducting a comprehensive analysis, we have ultimately decided to proceed with the latter approach, the single model for all themes. However, before we do so, it is important to consider the drawbacks of the first approach - using thirty separate models. Let us examine some of the cons of this approach.

- It is not a feasible solution to create a separate model for each theme because it requires a lot of time and resources for training and fine-tuning each model, and also for storage. However, having a specialized model might be necessary in specific use cases, such as answering medical questions, where domain knowledge plays a significant role.
- The data provided was not sufficient enough to fine-tune a separate model for each theme, even though we considered creating synthetic data to supplement it.
- To address the issue, we attempted to cluster themes together and build fewer models that could handle multiple themes in each cluster. During our experiment, we found that there was some similarity in the embedding space. The HDBSCAN algorithm resulted in only two clusters, but the dimension reduction technique using t-SNE showed 30 separate points, indicating that all themes were distinctly different from one another. Thus, clustering themes was not a viable solution.

As a result, we decided to go with a single, more generalized model that eliminates the problems mentioned above.

4) Using already answered questions: We have used the data released in the morning of the day of evaluation to fine-tune our model on the 30 new themes. This, along with the finetuned model for task-1 gave a significant improvement in the final F1 score based on exact match of answers. Using the fine-tuned model allowed us to get a match of 56% from an earlier exact answer match in 41.2% of the questions.(This was performed on the entire dataset that was provided on the day of evaluation) This gave us an approximate idea of the performance of our model. We were provided with a new set of question-answer pair on the day of evaluation. The dataset consisted of questions and answers, with the paragraph ids containing the paragraph that answers the question. The data only had 'true' values, meaning that all question-paragraph pairs had answers in the paragraphs. Thus we wanted a way in which we could use this data for the retrieval pipeline(task1 MiniLM). To address this, we added 'false' values to the dataset by randomly linking questions to paragraphs that did not contain the answers. This created a dataset of question-paragraph pairs with answers not present in the paragraph and provided us with a dataset to finetune our Task1 MiniLM model. We had 943 true cases and 940 false cases in the modified dataset.

II. Complete Pipeline

In this approach, we present a novel natural language processing (NLP) based pipeline for question answering. The pipeline inputs a question and a set of paragraphs and uses two state-of-the-art NLP models, BM 250KAPI and Mini LM, to produce accurate answers.

First, the input question and paragraphs are processed by the BM 250 KPI model, which uses advanced language modeling techniques to rank the paragraphs based on their relevance to the question. The top 10 ranked paragraphs are then selected for further processing.

Next, these top 10 paragraphs are combined with the input question and processed by the Mini LM model, which outputs a binary classification of "true" or "false" for each paragraph. If all outputs are "false," the pipeline returns -1 and terminates. If one of the outputs is "true," the corresponding question and paragraph are concatenated and sent through a finetuned Mini LM model specifically designed for question answering. This model outputs the start and end indices of the answer within the paragraph.

The results of our pipeline demonstrate the effectiveness of using advanced NLP techniques for question answering and the potential for further refinement and optimization of the pipeline. This research contributes to the ongoing effort to develop and improve NLP-based question answering systems.

III. Result and run time analysis

After extensive experiments, we chose MiniLM as our backbone due to its superior performance on both tasks and its smaller model size.

Model	Training Time
Task1_MiniLM	360
Task2_MiniLM	75

TABLE II: Total number of trainable parameters comparison

Class	Average Prediction time per question (sec)
Materialism	4.12
Federal_Bureau_of_Investigation	3.41
Human_Development_Index	3.40

TABLE III: Top-3 classes with highest average prediction time per question

Class	Average Prediction time per question (sec)
The_Times	1.23
Nanjing	1.30
DevRev	1.32

TABLE IV: Top-3 classes with lowest average prediction time per question

The training time analysis, including the top three classes with the highest and lowest prediction times, is presented below.

IV. Conclusion

The question answering pipeline uses MiniLM demonstrated decent accuracy in identifying answers to questions within relevant paragraphs. However, the retrieving mechanism could be improved for better results. The retrieval pipeline effectively used fine-tuned MiniLM and rankBM25, but more sophisticated retrieval methods could be considered to increase accuracy. Additionally, the deep learning models used in the pipeline could be further optimized through techniques such as model compression or pruning. Despite these areas for improvement, the submission provided valuable insights and demonstrated sufficiently par results.

V. Literature Survey

NLP is a rapidly evolving field, and there have been numerous research efforts in recent years to address a wide range of NLP tasks and problems. In this section, we will review some of the most relevant prior work that has been published on the topic of our hackathon project. Transformer networks have been shown to be very effective at natural language tasks, including question answering, and are often used in resource-constrained environments due to their efficiency and ability to be parallelized. For example, BERT (Bidirectional Encoder Representations from Transformers), a pre-trained language model, is particularly effective at handling long-range dependencies in language, and has been widely used in resource-constrained environments. For our problem, we found a number of relevant work in the field of question-answer

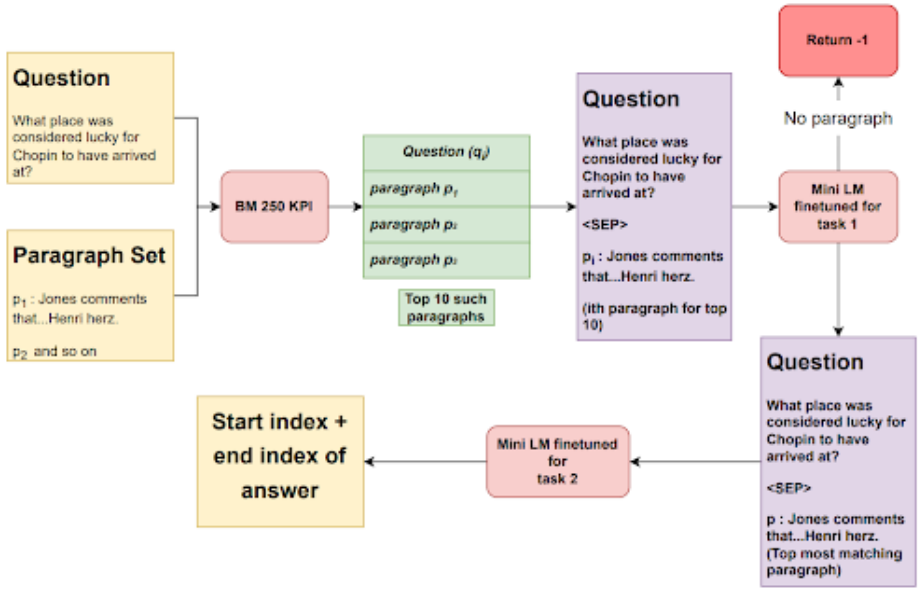

Fig. 2: Complete pipeline

generation (relevant to the second task), and in paraphrasing (relevant to data augmentation). While the vast majority were not used directly in our solution due to the hackathon guidelines, they provided valuable insight into the training process. We list them below.

- "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization" by Zhang et al. (2020): This paper introduces PEGASUS, a transformer-based language model for abstractive summarization. PEGASUS is trained using a new pre-training task called gap sentence prediction, in which the model is given a sentence with a randomly-masked span and must predict the words in the masked span. The authors show that PEGASUS outperforms previous state-of-the-art models on several abstractive summarization benchmarks
- "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension" by Lewis et al. (2020): This paper introduces BART, a transformer-based denoising autoencoder for pre-training sequence-to-sequence models. BART is trained to reconstruct a sentence from a corrupted version of the sentence, and is shown to achieve strong results on several natural language generation, translation, and comprehension tasks.
- Text Summarization with Pre-trained Encoders" by Liu et al. (2019): This paper investigates the use of pre-trained transformer encoders for text summarization. The authors show that fine-tuning a pre-trained transformer model on a small amount of labeled data can lead to strong performance on summarization benchmarks.
- T5 (Text-To-Text Transfer Transformer): This is a transformer-based language model developed by Google Research. T5 is trained to perform a wide range of natural language tasks using a single, unified framework, and has achieved strong results on a variety of benchmarks. (Raffel et al. 2019)
- DistilBERT: This is a smaller, faster version of BERT that has been trained to retain most of the performance of the original BERT model while being more computationally efficient. DistilBERT has fewer layers and attention heads than BERT, making it a good choice for resource constrained environments. (Sanh et al. 2019)

In addition, the paper EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks by Jason Wei, Kai Zou, was immensely helpful in the data augmentation process.

REFERENCES

[1] A. Trotman, A. Puurula, and B. Burgess, "Improvements to bm25 and language models examined," in *Australasian Document Computing Symposium*, 2014.

[2] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," 2020.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *CoRR*, vol. abs/1911.02116, 2019. [Online]. Available: http://arxiv.org/abs/1911.02116

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. [Online]. Available: http://arxiv.org/abs/1909.11942

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” 2019. [Online]. Available: https://arxiv.org/abs/1910.01108
[7] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization,” 2019. [Online]. Available: https://arxiv.org/abs/1912.08777