Technical Document Query System using Transformer Model-based Machine Reading Comprehension

Amit Kumar
AU Small Finance Bank Limited
Jaipur, India
amitjohn007@gmail.com

Friska Natalia
Information Systems Department
Universitas Multimedia Nusantara
Tangerang, Indonesia
friska.natalia@umn.ac.id

Sud Sudirman
School of Computer Science and Mathematics
Liverpool John Moores University
Liverpool, United Kingdom
s.sudirman@ljmu.ac.uk

Dhiya Al-Jumeily
School of Computer Science and Mathematics
Liverpool John Moores University
Liverpool, United Kingdom
d.aljumeily@ljmu.ac.uk

***Abstract*— Constructing a Question Answering system is a challenging task despite a significant amount of study that has been conducted in recent times on this topic. It is even more difficult to provide satisfactory responses to the inquiries raised by users in an organizational setting as opposed to in an informal setting. We present in this paper, the results of our study into the use of a transformer-based model in the development of a technical document query system with machine reading comprehension. Our method fine-tunes a pre-trained transformer model with hyperparameter optimization using a pre-processed training dataset and tested on a different dataset. We experimented using eight pre-trained models from seven different variations of the BERT transformer architecture including BERT, RoBERTa, XLM-RoBERTa, ELECTRA, ALBERT, MobileBERT, and MPNet using the SQuAD1.1 dataset for fine-tuning and the Oracle Knowledge Documentation for testing. We found that the ALBERT pre-trained model is the best model achieving 0.891, 0.950, and 0.882 performance when measured using the Exact Match, F1 score, and Confidence Score metrics - despite its relatively small model size.**

Keywords*— *Question Answering, Transformer model, Bidirectional Encoder Representations, Natural Language Processing, Chatbot system.

I. Introduction

A Question-Answering (QA) system is a computer program that provides acceptable responses to questions posed in natural languages such as English or Chinese, etc. Take for instance when a user queries, "Which team won the first cricket world cup?" it is anticipated that the QA system would respond with "West Indies". We will be able to access information more quickly with the assistance of a QA system. Constructing a QA system is a challenging task despite a significant amount of study that has been conducted in recent times on this topic. It is even more difficult to provide satisfactory responses to the inquiries raised by users in an organizational setting as opposed to in an informal setting. Automated chat assistants are becoming more popular among businesses as a means to manage conversations including technical help and customer support. However, these technologies are only able to effectively troubleshoot the problems on which they were trained, which presents a major difficulty for corporate QA procedures today. Some typical scenarios where question answering is being used are in customer and technical support.

In this paper, we present our methodology to develop a chatbot system to query a technical document using a transformer model-based machine reading comprehension system. The transformer model is developed by fine-tuning an existing pre-trained model. Several models were considered and evaluated on the Oracle technical documentation as a use case. The chatbot system is implemented using Python Flask and Webhooks. The final fine-tuned models are saved online for easy download and replication of results.

II. Literature Review

In the past few years, most of the research has been centered around enhancing the existing model performance on high-performance datasets available in the market. However, the interplay between context and question was modeled for the very first time using a model called the Bidirectional Enhanced Attention Flow (BiEAF) model [1]. This endeavor is a landmark achievement in this field of study. However, the BiEAF approach only takes into account the connection between sentences and it does not take into consideration the relationship within the phrases themselves [2]. A different approach called Knowledge-Based Question Answering has also been proposed. Its application in a chatbot framework whose goal is to enhance the framework's capacity to recognize and answer informal and natural language patterns through the use of well-structured relational information between entities is reported in [3]. A different solution was proposed in [4] by storing the lexical, semantic, and syntactic features of the words and phrases using distributional representations and using Convolutional Neural Network (CNN) to rate the possible responses. Several other machine learning techniques have been proposed to perform the feature classification task in this context including Support Vector Machine (SVM) [5], Recurrent Neural Network (RNN) [6], Deep Feature Matching [7], and Conditional Random Field (CRF) [8]. Many of the proposed methods here are reported to have achieved high performance on different datasets.

The transformer model in Natural Language Processing (NLP) is a novel architecture that aims to solve sequence-to-sequence understanding problems while handling long-range dependencies. The method is relatively new compared to the more established Long-Short Term Memory (LSTM) approaches. It relies on the self-attention mechanism to compute representations of its input and output without using sequence-aligned convolution networks. The encoder and decoder are the two systems that make up the transformer architecture. The input text is encoded by the encoders, and the decoder examines the encoded text in order to comprehend the contextual information that lies behind the sequence. Each encoder and decoder in the stack make use of an attention mechanism to evaluate each input in conjunction with every other input to balance the relative significance of the inputs. The encoder then works in conjunction with the decoder to construct the output sequence. The attention method in this case makes it possible to grasp the characteristics of the incoming text and dynamically highlight parts of it.

Different tasks require different uses of the encoder, the decoder, or both. Certain tasks, such as the QA task, the Masked Language Model (MLM) task, and the Next Sentence Prediction (NSP) task, require only the encoder part while others may require the decoder part or both. One of the most popular types of encoder-only transformer models is BERT, which stands for Bidirectional Encoder Representations [9]. The BERT architecture has given birth to many other BERT-like models such as RoBERTa [10], XLM-RoBERTa [11], ELECTRA [12], ALBERT [13], MobileBERT [14], and MPNet [15]. The original, or cased, BERT model uses 12 layers of transformers block with a hidden size of 768 with 12 self-attention heads and has around 110 million trainable parameters. Due to its huge size, the training process is quite computationally intensive. The QA, MLM, and NSP tasks are just a few of many different tasks in which training processes capture the bi-directional understanding of the sentences. The model used in MLM hides part of the values taken from the input and then uses the context to make an educated guess as to what the hidden (or missing) word is. As mentioned previously, the BERT model has been used to derive many other transformer models and in conjunction with other approaches. Many of these approaches are developed and tested on publicly available datasets including the Stanford Question Answering Dataset (SQuAD) [16], NII Testbeds and Community for Information Access Research (NTCIR) dataset [17], Semantic Textual Similarity Benchmark (STSB) dataset [18], Microsoft Research Paraphrase Corpus (MRPC) dataset [19], Sentences Involving Compositional Knowledge (SICK) dataset [20], and Microsoft Wikipedia QA (WikiQA) dataset [21]. A selection of some of the works in this regard and the datasets they are developed and tested on is provided in Table I below.

TABLE I. PREVIOUS STUDIES USING THE TRANSFORMER MODEL, THE DATASET THEY ARE DEVELOPED AND TESTED ON, AND REPORTED RESULT

Authors	Dataset used	Techniques	Reported Results
Day et. al. [5]	NTCIR-12 QA-Lab2 Japanese University entrance exams	Stanford NER and Stanford POS tagger from Stanford Core using NLP + SVM classifiers	1) Accuracy = 0.835 (on balanced dataset), 2) Accuracy = 0.90 (on imbalanced dataset)
Anhar et. al. [6]	Indonesian public corpora chat	1) RNN, 2) LSTM, and 3) Bi-LSTM	1) Technique #1 accuracy = 0.523, 2) Technique #2 accuracy = 0.871, 3) Technique #3 accuracy = 0.909
Vekariya et. al. [7]	STSB dataset, MRPC dataset, SICK dataset, and WikiQA dataset.	Word vector model, Versatile global T-max pooling, Deep LSTM, and Efficient DFM	1) Wikipedia QA dataset (Accuracy = 0.853, Recall = 0.812), 2) MRPC dataset (Accuracy = 0.873, F1 = 0.832), 3) SICK dataset (Accuracy = 0.881, Recall = 0.863), 4) STSB dataset (Accuracy = 0.832, Recall = 85.5)
Cai et. al. [22]	CCKS2018 dataset	word2vec + CNN + Bi-LSTM + Co-attention mechanism.	1) Precision = 0.860, 2) Accuracy = 0.864, 3) Recall = 0.872, 4) F1 Score = 0.866
Day et. al. [23]	Delta Reading Comprehension dataset	BERT + Q-EAT + Answer type classification model	1) Exact Match = 0.802, 2) F1 Score = 0.894
Wang et. al. [8]	Proprietary DieaseQuestion and DieaseBase datasets	BERT + Bi-LSTM layer + CNN layer + CRF layer	Accuracy = 0.824
Yang [2]	SQUAD dataset	Encoder-decoder mechanism with attention flow	1) Exact Match = 0.685 and 2) F1 Score = 0.777
Japa and Rekabdar [3]	Web question dataset	BERT + CNN	F1 Score = 0.564
Singh et. al. [4]	Text Retrieval Conference (TREC) QA dataset	(NLP methods of pattern matching and information retrieval) + CNN	1) Mean Average Precision (MAP) = 0.635 and 2) Mean Reciprocal Rank (MRR) = 0.657
Arora et. al. [24]	SQUAD dataset	1) GloVe + bidirectional-LSTM, 2) InferSent + XG Boost and Multinomial Logistic Regression	1) Technique #1 accuracy = 0.603, 2) Technique #2 accuracy = 0.660
Lan et. al. [25]	Chinese complex question and answer corpus	BERT-LSTM	F1 Score = 0.825
Sharath and Banafsheh [26]	Web-Questions dataset gathered from Google Suggest API, and the manually labeled answers from Amazon Mechanical Turk.	BERT+CNN+ Attention mechanism	F1 Score = 0.554
Xiao [27]	SQUAD dataset	Embedding layer + Transformer layer + Attention layer	1) Exact Match = 0.687 and 2) F1 Score = 0.779

Wang and Lu [28]	The NLPCC2018 KBQA task dataset	BERT + BiLSTM + CRF	F1 Score = 0.901
Zheng et. al. [29]	Wiki QA and a certain municipal power grid customer service system dialogue material knowledge base	BERT + LSTM_CGN + Attention mechanism	Accuracy = 0.754

As can be seen in the table, a significant amount of study has been conducted in recent times on this topic. The study is being carried out with the assistance of a variety of datasets, which are either publicly available or proprietary. In addition to this, the researchers employed sophisticated processes such as deep learning or machine learning strategies to construct their models. Additionally, Webhook and Flask are being used to implement some aspects of the models. In terms of performance measurement, accuracy, precision, recall, and F1 score are four of the most popular metrics used. All of these approaches are taken into consideration in the design of our proposed methodology, which is described in the next section.

III. Material and Method

An overview of the proposed methodology is provided as a flowchart in Figure 1. In general, the method involves fine-tuning a pre-trained transformer model with hyperparameter optimization using a pre-processed training dataset. The resulting fine-tuned model will then be tested on a different dataset and the performance will be measured and analyzed.

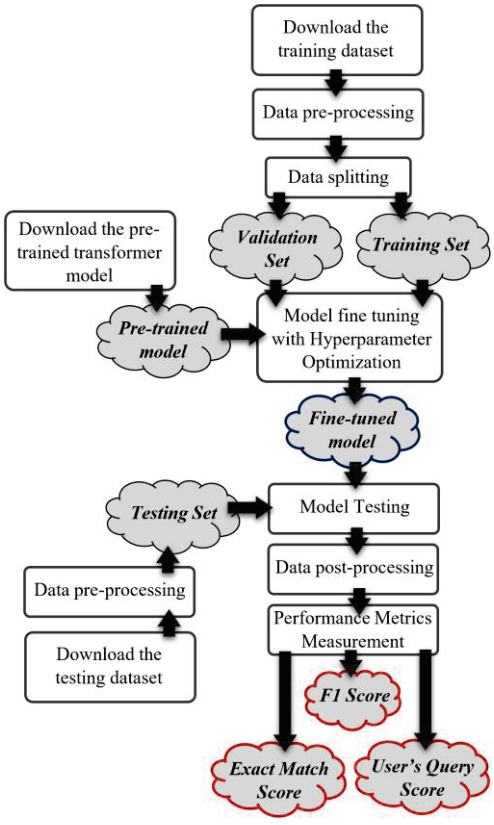

Fig. 1. An overview of the proposed methodology

For the training, the SQuAD1.1 dataset is used. This dataset was developed initially to test reading comprehension. The are several versions of this dataset, with SQuAD1.1 and SQuAD2.0 being the main ones. The questions and answer pairs in this dataset were formulated via the use of crowd-sourcing with Wikipedia serving as the foundation. The SQuAD 1.1 dataset comprises 107,785 question-answer pairs on 536 articles whereas the SQuAD2.0 dataset combines the questions and answers that are in SQuAD1.1 with an additional 53,775 unanswerable questions – i.e., questions whose answers do not exist in the provided context. The dataset has been used by many researchers and has been shown to provide a good benchmark since not only are systems required to provide answers to questions when it is feasible to do so, but they must also be able to discern when no response can be supported by the provided context and refrain from providing an answer. The Oracle technical documentation that will be used as the testing dataset contains information on products, industries, resources, customers, partners, developers, and events that provide an excellent use case for the developed methodology.

The input data for QA tasks may either be in the form of JSON files or a Python list of dictionaries that have been formatted appropriately. The structure of both forms is the same, for example, the input may be a string referring to a JSON file that contains a list of dictionaries, or it could be the list of dictionaries themselves. The most challenging aspect of this step is producing labels for the response to the question. These labels will indicate the beginning and ending locations of the tokens that match the answer inside the context. For this, we used a tokenizer to transform the text that is included in the input into entities that can be understood by the model. We provide our tokenizer with pairs of inquiry and context and it will correctly insert special tokens to build a sentence for us. The labels represent the indexes of the tokens that start and finish the response, and the model will be given the job of predicting one start and one end logits for each token in the input, in the following format: [CLS] Question [SEP] Answer [SEP].

Many of the samples that are included in the dataset have extremely lengthy contexts that will go above the maximum length that we have set. We deal with lengthy contexts by extracting multiple training features from a single sample and inserting a sliding window in between each of these features. By doing this, we can train the system on a more diverse set of circumstances. In our case, we set the length limit to 100 and use a sliding window with 50 tokens. An example label is shown in Figure 2 below.

```
'[CLS] To whom did the Virgin Mary allegedly appear in 1858 in Lourdes
France? [SEP] Architecturally, the school has a Catholic character.
Atop the Main Building\'s gold dome is a golden statue of the Virgin
Mary. Immediately in front of the Main Building and facing it, is a
copper statue of Christ with arms upraised with the legend " Venite Ad
Me Omnes ". Next to the Main Building is the Basilica of the Sacred
Heart. Immediately behind the basi [SEP]'

'[CLS] To whom did the Virgin Mary allegedly appear in 1858 in Lourdes
France? [SEP] the Main Building and facing it, is a copper statue of
Christ with arms upraised with the legend " Venite Ad Me Omnes ". Next
to the Main Building is the Basilica of the Sacred Heart. Immediately
behind the basilica is the Grotto, a Marian place of prayer and
reflection. It is a replica of the grotto at Lourdes, France where the
Virgin [SEP]'

'[CLS] To whom did the Virgin Mary allegedly appear in 1858 in Lourdes
France? [SEP] Next to the Main Building is the Basilica of the Sacred
Heart. Immediately behind the basilica is the Grotto, a Marian place of
prayer and reflection. It is a replica of the grotto at Lourdes, France
where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous
in 1858. At the end of the main drive (and in a direct line that
connects through 3 [SEP]'

'[CLS] To whom did the Virgin Mary allegedly appear in 1858 in Lourdes
France? [SEP]. It is a replica of the grotto at Lourdes, France where
the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in
1858. At the end of the main drive (and in a direct line that connects
through 3 statues and the Gold Dome), is a simple, modern stone statue
of Mary. [SEP]'
```

Fig. 2. An example of output labels produced by the data pre-processing stage.

In this example, this particular information from the sample has been separated into four distinct inputs. As we can see, each of these inputs has the same question but different contexts. It is important to take note of the fact that the correct response, i.e., "Bernadette Soubirous" does not appear in any of the inputs but the last one. This provides an example where the training instances contain a response that is not part of the context which will need to be fixed manually. We fix these cases by setting the text start position equal to the finish position which equals 0. A similar solution is applied in the event of the answer being cut short, leaving us with just the beginning (or the conclusion) to work with. However, if it is possible to find the correct response in its entirety inside the context, the labels will be the index of the token that corresponds to the beginning of the answer and the index that corresponds to the end of the answer.

After our training data has been thoroughly pre-processed, we can finally aggregate it into a function that can be applied to the whole dataset. Given that most contexts will be large (and the associated samples will be split into multiple features), we do not find it necessary to perform any dynamic padding. The SQuAD1.1 dataset includes several questions that have been modified by having blank spaces added to either the beginning or the end of the question. Since we do not contribute any additional information to the dataset, we eliminated those additional spaces because there is no use in keeping them.

The subsequent data-splitting stage divides the pre-processed dataset into training and validation sets by a 70:30 ratio, respectively. The training set is used exclusively to fine-tune the model and the validation set is used to determine when the process should stop. The fine-tuning stage is performed by first replacing the last fully connected output layer of the pre-trained model with a new output layer with a randomly initialized set of weights. The modified model is then trained on the training set using a set of hyperparameter values. At certain intervals, the model is saved and the training loss and validation loss are calculated. If the validation loss is less than the training loss and a pre-determined low value, then the model is considered as the final fine-tuned model, otherwise, the process is repeated after making adjustments to the hyperparameter values. This fine-tuning stage is illustrated in Figure 3.

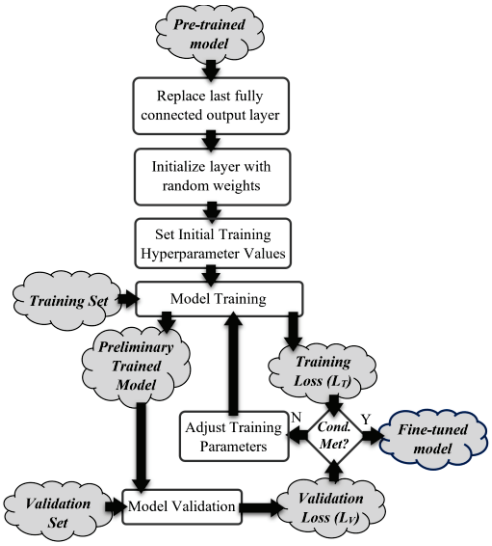

Fig. 3. A flowchart detailing the model fine-tuning stage.

The final fine-tuned model will be tested on a testing set derived from a different set of data, which is the Oracle Knowledge Documentation [30]. We perform the same data pre-processing step on this training dataset to prepare them for the subsequent model testing stage.

When tested, the model generates logits that correspond to the starting and ending positions of the answer in the input IDs as its output for each query. The raw output has to undergo several post-processing steps, similar to those in the pre-processing stage we described previously to ensure the integrity of the returned results. These steps are carried out by removing any start and end logits that correspond to tokens that are outside the context range, removing any responses that have zero length, and removing any responses that are longer than the maximum value (set to 30 characters long).

Three metrics are used when evaluating the model performance, they are Exact Match (*EM*), F1 Score (F_1), and Confidence Score (*CS*). The Exact Match metric is calculated as an average of a binary scoring system that produces either one if the model output is identical to the expected output, or zero otherwise. The F1 score is calculated by comparing the number of common words in the model output and the expected output. Two measurements need to be calculated before the F1 Score can be determined, they are Precision (*P*) and Recall (*R*). Precision is the ratio of the number of shared words to the total number of words in the model output whereas Recall is the ratio of the number of shared words to the total number of words in the expected output. The F1 Score is then calculated as:

$$F_1 = 2 \times \frac{P.R}{P+R} \tag{1}$$

The calculation of the CS metric uses the probability of start and end logits produced by the model derived from the expected outputs. A Soft Max function is used to transform the start and end logits into probabilities. These values are then multiplied to get the overall score for each pair of tokens. In practice, the model produces multiple candidate answers with varying CS values. The one with the highest value is considered the best and selected as the output of the model. This highest CS value is the one recorded for this metric. A high CS value (> 0.9) typically corresponds to a near-exact

match of the expected output and a low value (< 0.3) signifies low confidence that the produced answers relate to the expected output.

IV. Implementation and Experimental Results

The proposed methodology is implemented in Python 3.6 with Tensor Flow 2.0 on Jupyter Notebook hosted by Google Colab. The training datasets, pre-trained transformer models, and data tokenizer function are obtained via the Hugging Face platform [31]. Eight pre-trained models from seven different variations of the BERT transformer architecture are used. The seven architecture variations are the original, or cased, BERT [9], RoBERTa [10], XLM-RoBERTa [11], ELECTRA [12], ALBERT [13], MobileBERT [14], and MPNet [15]. The description of the eight pre-trained models including the dataset on which they were pre-trained is given in Table II. The rightmost column of this table has the URL where the fine-tuned models can be downloaded and tested.

TABLE II. Description of the Eight Pre-trained Models and the URL Containing the Fine-tuned Models

Transformer Models	Pre-trained model description	URL of the final fine-tuned model
CasedBERT (SQuAD2.0)	A cased BERT model pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/simplebert-base-finetuned-squad
CasedBERT (BookCorpus Wiki)	A cased BERT model pre-trained on BookCorpus, a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables, and headers).	https://huggingface.co/amitjohn007/bert-finetuned-squad
RoBERTa	A base RoBERTa model pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/roberta-base-finetuned-squad
XLM-RoBERTa	A multilingual large XLM-RoBERTa pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/xlm-roberta-base-finetuned-squad
ELECTRA	A base ELECTRA model pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/electra-finetuned-squad
ALBERT	A large ALBERT model pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/albert-finetuned-squad
MobileBERT	A thin version of the cased BERT model pre-trained on SQuAD2.0 dataset	https://huggingface.co/amitjohn007/second-mobil-bert-finetuned-squad
MPNet	A base Microsoft MPNet model pre-trained on a COVID-19 dataset.	https://huggingface.co/amitjohn007/mpnet-finetuned

Each model's fine-tuning process takes a little more than an hour on the Google Colab Pro version. It is also worth noting that at the end of each epoch, a snapshot of the model and its training loss is stored in the Hugging Face repository. The values of the training loss of each model after the first, second, and third epoch are shown in Table III.

TABLE III. Training Loss After Each Epoch

Epoch	1	2	3
BERT (SQuAD2.0)	0.61	0.40	0.27
BERT (BookCorpusWiki)	1.27	0.78	0.57
RoBERTa	0.74	0.55	0.42
XLM-RoBERTa	0.66	0.46	0.29
ELECTRA	0.57	0.38	0.23
ALBERT	0.38	0.17	0.05
MobileBERT	0.64	0.53	0.46
MPNet	1.05	0.73	0.59

The performance of the fine-tuned models as measured using Exact Match, F1 score, and Confidence Score metrics on the test dataset is shown in Table IV. For analysis purposes, we also include the model size, in megabytes, in the rightmost column.

TABLE IV. The Performance of the Fine-tuned Models

Metrics	EM	F1 Score	CS	Size
BERT (SQuAD2.0)	0.799	0.881	0.812	431
BERT (BookCorpusWiki)	0.810	0.887	0.626	431
RoBERTa	0.855	0.921	0.627	497
XLM-RoBERTa	0.858	0.926	0.680	2240
ELECTRA	0.874	0.940	0.594	1340
ALBERT	**0.891**	**0.950**	**0.882**	823
MobileBERT	0.814	0.893	0.583	**93**
MPNet	0.858	0.923	0.505	438

From the table, we can see that the ALBERT model produces the best performance in all three metrics, despite having a relatively small model size (823 MB compared to the smallest i.e., 93 MB and the largest, i.e., 2240 MB).

To finish off, we show here the query chatbot system that was implemented using Webhooks in Python Flask. Flask is a microweb framework written in Python whereas Webhooks is an HTTP-based callback function that allows lightweight, event-driven communication between two application programming interfaces (API). A screenshot showing an example user interface of the chatbot is shown in Figure 4.

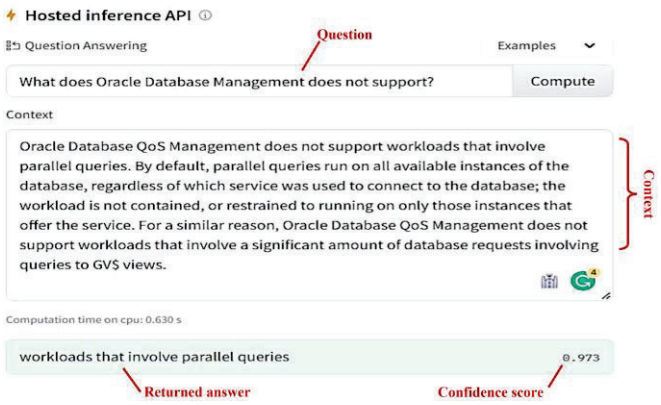

Fig. 4. An example screenshot of the developed chatbot's user interface.

V. Conclusion

We have presented in this paper the results of our study into the use of a transformer-based model in the development of a technical document query system with machine reading comprehension. Our method fine-tunes a pre-trained transformer model with hyperparameter optimization using a pre-processed training dataset and tested on a different dataset. Eight pre-trained models from seven different variations of the BERT transformer architecture are used. The seven architecture variations are the cased BERT, RoBERTa, XLM-RoBERTa, ELECTRA, ALBERT, MobileBERT, and MPNet. The fine-tuning process uses the SQuAD1.1 dataset whereas the testing is performed using the Oracle Knowledge

Documentation. We found that the ALBERT pre-trained model is the best model to use when its performance is measured using the Exact Match, F1 score, and Confidence Score metrics - despite its relatively small model size.

REFERENCES

[1] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv Prepr. arXiv1611.01603*, 2016.

[2] Y. Yang, "BiEAF: An Bidirectional Enhanced Attention Flow Model for Question Answering Task," in *2021 2nd International Conference on Information Science and Education (ICISE-IE)*, 2021, pp. 344–348.

[3] S. S. Japa and B. Rekabdar, "Memory Efficient Knowledge Base Question Answering with Chatbot Framework," in *2021 IEEE Seventh International Conference on Multimedia Big Data (BigMM)*, 2021, pp. 33–39.

[4] D. Singh, K. R. Suraksha, and S. J. Nirmala, "Question Answering Chatbot using Deep Learning with NLP," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2021, pp. 1–6.

[5] M.-Y. Day and C.-C. Tsai, "A study on machine learning for imbalanced datasets with answer validation of question answering," in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, 2016, pp. 513–519.

[6] R. Anhar, T. B. Adji, and N. A. Setiawan, "Question classification on question-answer system using bidirectional-LSTM," in *2019 5th International Conference on Science and Technology (ICST)*, 2019, vol. 1, pp. 1–5.

[7] D. V Vekariya and N. R. Limbasiya, "A novel approach for semantic similarity measurement for high quality answer selection in question answering using deep learning methods," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 518–522.

[8] X. Wang and Z. Wang, "Question answering system based on diease knowledge base," in *2020 IEEE 11th international conference on software engineering and service science (ICSESS)*, 2020, pp. 351–354.

[9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," no. 1, 2019.

[11] 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," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 8440–8451, 2020.

[12] C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," *Iclr 2020*, pp. 1–18, 2020.

[13] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: a Lite Bert for Self-Supervised Learning of Language Representations," *8th Int. Conf. Learn. Represent. ICLR 2020*, pp. 1–17, 2020.

[14] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: A compact task-agnostic BERT for resource-limited devices," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 2158–2170, 2020.

[15] K. Song, X. Tan, T. Qin, J. Lu, and T. Y. Liu, "MPNet: Masked and permuted pre-training for language understanding," *Adv. Neural Inf. Process. Syst.*, vol. 2020-December, no. NeurIPS, pp. 1–14, 2020.

[16] Stanford University, "The Stanford Question Answering Dataset (SQUAD)," 2023. [Online]. Available: https://rajpurkar.github.io/SQuAD-explorer/. [Accessed: 24-Oct-2023].

[17] NTCIR, "NII Testbeds and Community for Information Access Research," 2016. [Online]. Available: https://research.nii.ac.jp/ntcir/ntcir-12/data.html. [Accessed: 24-Oct-2023].

[18] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao, "Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization," *arXiv Prepr. arXiv1911.03437*, 2019.

[19] Microsoft, "Microsoft Research Paraphrase Corpus," 2016. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=52398. [Accessed: 24-Oct-2023].

[20] L. Zhang, S. R. Wilson, and R. Mihalcea, "Multi-label transfer learning for multi-relational semantic similarity," *arXiv Prepr. arXiv1805.12501*, 2018.

[21] Microsoft, "Microsoft Research WikiQA Corpus," 2016. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=52419. [Accessed: 24-Oct-2023].

[22] L.-Q. Cai, M. Wei, S.-T. Zhou, and X. Yan, "Intelligent question answering in restricted domains using deep learning and question pair matching," *Ieee Access*, vol. 8, pp. 32922–32934, 2020.

[23] M.-Y. Day and Y.-L. Kuo, "A study of deep learning for factoid question answering system," in *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*, 2020, pp. 419–424.

[24] R. Arora, P. Singh, H. Goyal, S. Singhal, and S. Vijayvargiya, "Comparative question answering system based on natural language processing and machine learning," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, 2021, pp. 373–378.

[25] J. Lan, W. Liu, Y. Hu, and J. Zhang, "Semantic Parsing and Text Generation of Complex Questions Answering Based on Deep Learning and Knowledge Graph," in *2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE)*, 2021, pp. 201–207.

[26] J. S. Sharath and R. Banafsheh, "Conversational question answering over knowledge base using chat-bot framework," in *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, 2021, pp. 84–85.

[27] Y. Xiao, "A transformer-based attention flow model for intelligent question and answering chatbot," in *2022 14th International Conference on Computer Research and Development (ICCRD)*, 2022, pp. 167–170.

[28] H. L. Wang and X. X. Lu, "Question Answering System with Enhancing Sentence Embedding," in *2022 11th International Conference of Information and Communication Technology (ICTech))*, 2022, pp. 521–524.

[29] C. Zheng, Z. Wang, and J. He, "BERT-Based Mixed Question Answering Matching Model," in *2022 11th International Conference of Information and Communication Technology (ICTech))*, 2022, pp. 355–358.

[30] ORACLE, "Oracle Knowledge Documentation," 2023. [Online]. Available: https://www.oracle.com/technical-resources/documentation/knowledge-documentation.html. [Accessed: 13-Oct-2023].

[31] Hugging Face, "Hugging Face. The AI community building the future. The platform where the machine learning community collaborates on models, datasets, and applications.," 2023. [Online]. Available: https://huggingface.co/. [Accessed: 19-Oct-2023].