BIRD-QA: A BERT-based Information Retrieval Approach to Domain Specific Question Answering

Yuhao Chen
School of Computing
Queen's University
Kingston, Ontario, Canada
yuhao.chen@queensu.ca

Farhana Zulkernine
School of Computing
Queen's University
Kingston, Ontario, Canada
farhana.zulkernine@queensu.ca

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671523

***Abstract*—During recent years, Question Answering (QA) systems have been widely used in many industries to provide round the clock online services to consumers from all over the world. The importance of such services became more evident during the pandemic in online medical services, education, training, marketing, system support and administration. Most of the existing systems apply simple rule-based QA strategy. Human-defined rules are used to apply pattern matching for extracting information from a given data or knowledge base to generate responses to user queries. However, rule-based pattern matching techniques are not intelligent enough to understand the context of the question to always generate appropriate responses and are static. In this work, we explored different data preprocessing strategies and BERT-style pre-trained models to build an information retrieval (IR)-based Domain specific QA framework named BIRD-QA, and created a domain specific knowledge base using website data of a university department. We implemented multiple variations of extended BERT and ALBERT-base models and validated our framework on reading comprehension task using the Stanford Question Answering Dataset (SQuAD) 1.1 and 2.0 datasets. Our extended ALBERT-based model achieved 75.4% Exact Match (EM) score and 78.8% F1 score. We also present a small feasibility test of our framework for departmental QA using data from a university website.**

Keywords — information retrieval, SQuAD2.0, question answering

I. Introduction

In recent years, because of the rapid advancement of Artificial Intelligence (AI), human-like robotic systems have been created and deployed in a wide range of areas, such as e-commerce, health care, education, and so on [1][2][3]. A Question Answering (QA) or Chatbot system is such an intelligent conversation agent that is able to give feedback and communicate with users [4]. Search engines such as Google and Bing enable efficient query of documents from a large number of domains, but do not provide specific answers from within a document or from an inconspicuous corner of a website. In contrast, QA systems are domain-specific and require knowledge extraction from a variety of non-web sources [5]. Rather than just ranking the most suitable web links, the QA system can give a specific answer in natural language.

During the recent pandemic, online text and voice-based QA services have become extremely popular [6]. However, most online QA services recently are either based on a rule-based system that is able to match the question to an answer from a prebuilt QA pairs database [7] or a list that links Frequently Asked Questions (FAQ) with corresponding answers [8]. Users often get a negative experience, because answers are unsatisfactory or direct users to other web pages, and the system takes a long time to retrieve the desired answers. In the worst case, the system fails to find an answer and directs the query to a human agent. Generally, service companies contain terabytes of data, and it is almost impossible to encode all the rules with the help of domain experts to create rule-based QA systems that can provide answers to all possible user queries.

Compared to the rule-based QA system, an information retrieval (IR) based QA system applies a variety of data processing, machine learning, pattern matching and other deep learning techniques to extract the answer from a text corpora or information sources [30]. IR based techniques suffer from the absence of definitive information resulting in less accuracy and increased inference time for real-time conversations but can help in finding answers that are not predefined. Although the recent deep learning techniques have the potential to learn to answer questions not existing in predefined question-answer pairs with much greater accuracy [10][11][18] compared to the traditional keyword-based QA methods, the requirement of high computational power makes this approach difficult to be realized in a real-time scenario [9]. Researchers have been exploring other techniques such as implementing a cache QA layer to store the frequently asked information and adding them periodically to the FAQ [31]. However, an efficient deep learning model to retrieve answers from big data would still be needed to create the new question-answer pairs. Large companies can use high priced GPUs to speed up model training and answer prediction for providing real-time knowledge or support services. However, for smaller organizations, it is still an open problem to develop an efficient QA framework for domain specific QA.

In this research we address the problem of developing an efficient domain specific QA framework. We started with exploring the state-of-the-art pretrained models such as BERT [10] and ALBERT [11], which have shown great success in reading comprehension tasks. Context-based language processing and creating an effective embedding of large text corpora is essential for QA tasks. We extended selected multiple BERT-base models, both high performance and less complex, using four different approaches to validate their effectiveness. We applied two different strategies for processing long sequence of text input. The extended models were validated for the reading comprehension task using the Stanford Question Answering Dataset (SQuAD) 1.1 and 2.0 datasets [21][22]. Our contribution of the study is the creation of a domain specific knowledge base using the website data of a university specifically of one department, and illustration of the proposed BIRD-QA framework in domain specific QA. Our extended fine-tuned ALBERT-base model for domain specific QA achieved the best performance, which is 75.4% Exact Match (EM) score and 78.8% F1 score.

The rest of the paper is organized as follows. Section II describes background concepts and relevant work. Section III presents an overview of the solution, dataset, and data

preprocessing strategy. Section IV illustrates the process of model development and validation. Finally, we conclude the paper, and state some future work directions in Section V.

II. Background And Related Work

A. Evolution of QA Systems

Natural Language Processing (NLP) is a research area that is trying to find the best way to teach machines to understand and encode linguistic features in order to understand natural language from written text or speech. It is able to perform specific language-based tasks such as next word prediction successfully. QA is one of those tasks that requires machines to answer users' questions, which proved to be very difficult due to three major challenges [12]: a) how to teach computers to think and analyze, b) how to represent the meaning of input words, and c) how to teach computers to understand and memorize world knowledge. Therefore, a much simpler rule-based QA system that does not require the ability of human-like thinking and understanding was developed. One of the most famous and earliest examples is ELIZA Chatbot which could discuss psychological issues [13]. However, this kind of chatbot was limited by the rules or needed a large predefined question and answer pairs.

After several years, intent detection and entity recognition with machine learning was developed. With the development of machine learning algorithms, rule-based QA systems started applying intent detection and entity recognition methods [14][15]. The system tried to retrieve keywords from the user input and find intents or category of those keywords by machine learning. Although pre-defined question and answer pairs were still used for QA, it was limited by the amount of structured data available to find the answer.

B. Deep Learning-based QA Systems

Recent advancements in the internet and big data requires a large FAQ to satisfy user queries and it is difficult to capture all possible user queries and define question-answer pairs for a rule-based QA system, With great achievements in the field of NLP and deep learning, the advanced QA systems are now able to apply generation-based or information retrieval-based QA [16] approaches, which do not require pre-defined question-answer pairs.

Generation-based QA systems automatically compose new answers to given questions based on a given corpora. The most common approach to create generation-based QA systems is to use machine translation methods, which match each input question to the answer through a Seq2Seq model [17]. By contrast, an information retrieval-based QA system tries to find the locations of most relevant answers for a set of questions in different documents of a given text corpora through deep learning [30]. The deep learning models in this kind of systems emphasize on learning the skills to find an answer within a document rather than learning to match simple word-to-word expressions, which allows it to find answers to questions from documents outside of the training dataset. Since the generation-based QA systems compose new answers, it is difficult to validate the correctness of the answers. Also, the system may generate meaningless answers. Compared to the generation-based systems, IR-based QA systems are therefore, much more controllable. These systems only retrieve answers from the documents accessible to the system through a given text corpora.

C. Related Work

QANET [54] was the first QA system that used convolutional neural networks with self-attention instead of RNN-based models to extract both local and global features of texts. Overall, the model consisted of five major components: a) Input Embedding Layer, which was used to concatenate GloVe word embedding and convolutional character embedding; b) Embedding Encoder Layer, which used a similar architecture as the Transformer encoder [28] but added multiple convolution blocks before the multihead attention block to generate a global contextualized representation; c) Bidirectional Attention layer which implements text processing in both forward and the backward directions; d) Model Encoder Layer for further feature extraction, and e) Output Layer for predicting the probability of the start and end positions of an answer in the context. QANET was as 3x to 13x faster than RNN-based models in training and was able to achieve significantly better performance for the same training time.

Because of the rapid development of BERT-style model, the performance of IR-based QA systems has shown a significant improvement [10][18][11]. BERT, proposed by Devlin et al. [10], is a language model that used Transformers [28] instead of LSTM. Different from Radford et al.'s GPT [29] that used unidirectional Transformers to train a language model, Devlin et al. introduced a new idea on training a Bidirectional Encoder Representation from Transformer (BERT) [10]. They achieved bidirectional representation by training a Masked Language Model (MLM) instead of using a shallow concentration of independently trained left-to-right and right-to-left language models. They also trained the model for another unsupervised task called Next Sentence Prediction (NSP) jointly for the model to learn sentence relationships. After BERT was introduced, it set new state-of-the-art results on 11 NLP tasks, including question answering and text classification [10].

ALBERT is a light version of the BERT-style model, which achieved a comparable performance as BERT with less computational and architectural complexity [11]. It achieved this by three major adjustments in the architecture of BERT: a) factorized embedding parameterization, which separates the size of hidden layers from the size of embedding; b) cross-layer parameter sharing, which enables parameter sharing of each layer with the other layers to reduce the GPU usage, and c) inter-sentence coherence loss, which is used for sentence order prediction instead of next sentence prediction.

III. Overview of the Solution

The aim of this study was to develop a deep learning model to create an IR-based QA system given a knowledge base (KB) to extract the answer to a question. To maintain the accuracy and reliability, we wanted the system to respond with answer not found instead of providing a wrong answer. Therefore, we experimented with different state-of-the-art BERT-style pre-trained language models such as BERT [10] and ALBERT [11] on QA tasks to validate their effectiveness and usability for real time question answering. Fig. 1 shows our implementation framework. Our system is able to retrieve the answers from our collected data (referenced documents) by a deep learning model. If the question cannot be answered by referenced documents, it will return "Sorry, I don't know" for feedback. The framework, specifically the deep learning models, are validated using benchmark datasets that are

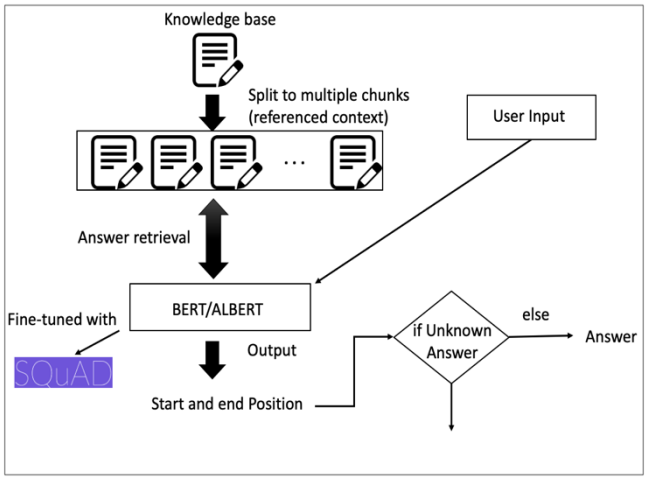

Fig. 1. The Architecture of our IR-based QA System

commonly used for QA tasks. We also created a KB to demonstrate how our models and framework can be used in a real-life use case scenario.

TABLE I. RESULTS OF SQUAD1.1 AND 2.0 DEVELOPMENT (DEV) SET

Model	*SQuAD1.1 (EM/F1)*	*SQuAD2.0 (EM/F1)*
QANet [19]	73.6/82.7	N/A
BiDAF + Self Attention [20]	67.7/77.3	N/A
BERT base [10]	80.8/88.5	N/A
ALBERT base [11]	82.3/89.3	N/A

- N/A – was not reported in published work

Table I shows the performance of 4 different QA models namely QANet [19], BiDAF [20], BERT [10], and ALBERT [11] on reading comprehension tasks. Thus, in this research, we use the BERT-style model as the backbone of our QA system which is pre-trained with Wikipedia articles and fine-tuned with SQuAD.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Fig. 2. Sample question answering pairs in SQuAD 1.1 [21]

The key components of the framework are described in the following subsections.

A. Dataset

We train and evaluate our deep learning models on two datasets. One is the Stanford Question Answering Dataset 1.1 (SQuAD 1.1) [21], which is a reading comprehension task that contains 100,000+ questions from Wikipedia articles. Each answer to these questions is a segment of text from the corresponding passage. As shown in Fig. 2, each sample in SQuAD 1.1 consists of three main components: a referenced paragraph, questions, and their answers. The task is about retrieving the answer to a given question from the referenced paragraph. So for the deep learning model, the input is the referenced paragraph and question, and the predicted label is the answer phrase inside the referenced paragraph.

We also used version 2.0 of the SQuAD dataset that not only contains questions from SQuADv1.1, but also includes

Article: Endangered Species Act
Paragraph: "*... Other legislation followed, including the Migratory Bird Conservation Act of 1929, a 1937 treaty prohibiting the hunting of right and gray whales, and the Bald Eagle Protection Act of 1940. These later laws had a low cost to society—the species were relatively rare—and little opposition was raised.*"

Question 1: "*Which laws faced significant opposition?*"
Plausible Answer: *later laws*

Question 2: "*What was the name of the 1937 treaty?*"
Plausible Answer: *Bald Eagle Protection Act*

Fig. 3. Sample question answering pairs in SQuAD 2.0 [22]

over 50,000 unanswerable questions created by crowd-workers [22]. This dataset is much more challenging because the system should consider two situations: a) when an answer is supported by the referenced paragraph, and b) when no answer is supported by the referenced paragraph. Compared with SQuAD 1.1, SQuAD 2.0 contains additional labels for the unanswerable questions. SQuAD 2.0 consists of 5 main components: a referenced paragraph, questions, answers, impossible to answer or not, and a plausible answer. For an answerable question, the dataset contains an answer from the referenced paragraph and "is impossible: false", and the plausible answer contains an empty string. Fig. 3 shows a sample data from the SQuAD 2.0 and two unanswerable questions that return incorrect answers as plausible answers as extracted from the referenced paragraph, where answers contain empty strings and "is impossible: true". A plausible answer is provided in the dataset, which is something of the same type as what the question asks for [22], but does not answer the question exactly. We used SQuAD as the dataset to train and test our models in order to learn the relationship between the questions and the referenced paragraphs to retrieve the answer from a paragraph, and also detect whether the question is answerable or not. However, for the use case scenario, we did not use the referenced Wikipedia articles used for SQuAD. Instead, we created our own KB to demonstrate real life machine conversation for a university information services use case scenario as explained later in this section.

Our goal for the QA system is to detect whether a question is answerable or not, and only answer the questions that are possible to be answered from the referenced paragraphs and otherwise return "Sorry, I don't know". Therefore, SQuAD

2.0 [22], which includes both answerable and unanswerable questions, is the best choice for us to train our models with. However, due to the high GPU RAM requirement and slower training speed, we decided to experiment our ideas on the SQuAD 1.1 [21] first and then choose the best model to evaluate on SQuAD 2.0.

B. Data Preprocessing

For SQuAD 1.1, we read the JSON data file and extract the referenced context, question text, answer text, all possible answers, and the answer start position for each data sample. We use each answer text and its start position to compute the answer end position and then tokenize the data samples to the features that can be processed by BERT-style models. We use the tokenizer that is based on WordPiece [23], or SentencePiece [24] for different BERT models. During tokenization, "[CLS]" tokens are added to the beginning of each sample, and "[SEP]" tokens are used to separate the question and the referenced context. We also restrict the sequence length of our inputs by setting the maximum sequence length to 384 [10]. For those inputs that are shorter than the max sequence length parameter, we pad it with '0' to set its length to 384. For those inputs that are longer than the max sequence length parameter, we apply two strategies as described below and demonstrated in Fig. 4.

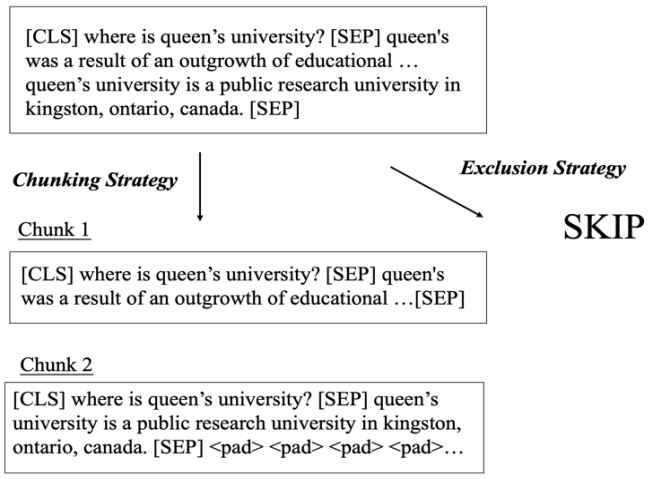

Fig. 4. Two strategies used in BIRD-QA for processing long sequence of text in SQuAD dataset.

Exclusion Strategy: We exclude all the input samples in training and evaluation datasets that have a long sequence. Only a small percentage of data samples have a longer sequence, so this will not have much influence on training and evaluation.

Chunking Strategy: We apply chunking, that is splitting the referenced context into multiple chunks. There are some overlapping information between these chunks, and therefore, each chunk can be shorter than the max sequence length.

The result of the tokenization process consists of four components: a) the input ids, which is a vector that stores the indices of input sequence tokens in the vocabulary; b) attention masks, which use 1 to represent the input text, and 0 to represent the padding, as used by BERT-style models to avoid computation of attention on padded tokens; c) token type ids, which store the position information of a question and the referenced context, and d) the start and end positions, which are converted to correspond to the word indices. An example of the feature outputs is shown in Fig. 5.

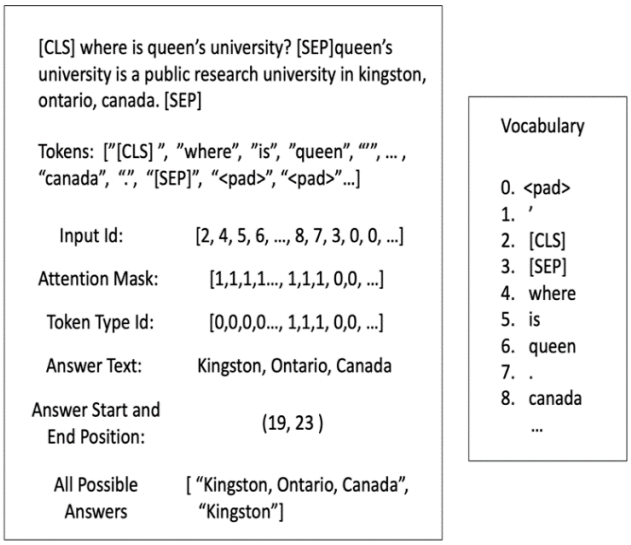

Fig. 5. Example of an output from tokenization

For SQuAD 2.0, we use the same data preprocessing strategy as used for SQuAD 1.1 with some additional steps. For the questions that are impossible to be answered, we use 0 to represent the start and end positions of the answer from the referenced context. We also use an additional vector to record whether a question from each sample is answerable or not: 1 means that it is impossible to be answered, and 0 means that it is possible to be answered. We ignore the plausible answer, because we do not want our model to provide a meaningless answer, and we would like it not to provide an answer for those questions that are impossible to be answered.

C. Model Extension and Training Strategies

We applied two approaches to extend pretrained BERT-style language models for downstream tasks such as language comprehension and QA. One is the ***feature-based*** approach, which freezes the weights of the pretrained language model to generate static contextualized embeddings such as ELMo [25]. Another one is the ***fine-tuning*** approach, which makes the weights of the pretrained language model trainable to generate dynamic contextualized embeddings based on new tasks such as BERT [10]. During fine-tuning, BERT-style models such as BERT, RoBERTa [18], and ALBERT [11] only need an additional dense layer for prediction to get the state-of-the-art performance. It is unnecessary to add a complex layer architecture after BERT-style models.

D. Knowledge Base

For demonstrating the feasibility of our system for real life application scenario, we use a web scraping[1] script to collect unstructured data from the departmental website as the KB to retrieve the question answers. We convert the HTML files from the departmental website to the same format as the Stanford Question Answering Dataset (SQuAD) to create the reading comprehension dataset to train the models for QA.

IV. Model Development and Validation

We explore multiple strategies and approaches to develop BERT-style models including BERT and ALBERT on the

[1]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

SQuAD machine reading comprehension benchmark dataset. For SQuAD 1.1, we only applied exclusion strategy for long sequences. For SQuAD 2.0, we applied both chunking strategy and exclusion strategy. During evaluations, we matched all the predicted start and end positions with the corresponding answer phrases from each of referenced context. We applied two different metrics to evaluate our predictions. a) Exact match (EM) score [21], which measures the percentage of predictions that match one of the labeled answers exactly, and b). Macro-averaged F1 score [21], which measures the average overlap between the predicted and labeled answers. For EM score, if the prediction matches one of the ground truth answers exactly, then 1 is added to the score, and otherwise not. For F1 score, even if the prediction does not match the ground truth answers exactly, we award some points based on the amount of overlap of the predicted answer with the ground truth answer. We propose several deep learning models which require a large computational power. In order to make the experiments much more efficient, we first evaluate the models on SQuAD1.1 which contains less data and only answerable questions, and then choose the best model to evaluate on SQuAD2.0 that contains more data and both answerable and unanswerable questions.

A. Experimental Setup

All experiments described in this paper were completed on an Ubuntu Linux 18.02 Server. It has a NVIDIA Tesla V100 GPU with 32G video memory.

B. Model Development

In experimentation, we fixed the maximum sequence length to 384, and set 48 for training batch size. Exclusion strategy was applied to SQuAD 1.1, and both exclusion and chunking strategies were applied to SQuAD 2.0. We explored two different techniques to train our BIRD-QA models for near real time departmental question answering use case scenario. For feature-based training approach, we recorded the best averaged EM and F1 scores after 10 epochs. For fine-tuning approach, we record the best model after 3 epochs.

In the following subsections, we describe four different approaches to extending BERT and ALBERT-base models based on the two model extension approaches, feature-based and fine-tuning based, as described in Section III.C. For each of the four approaches, we implemented multiple models, which are explained below along with the baseline models with a discussion of the experimental results.

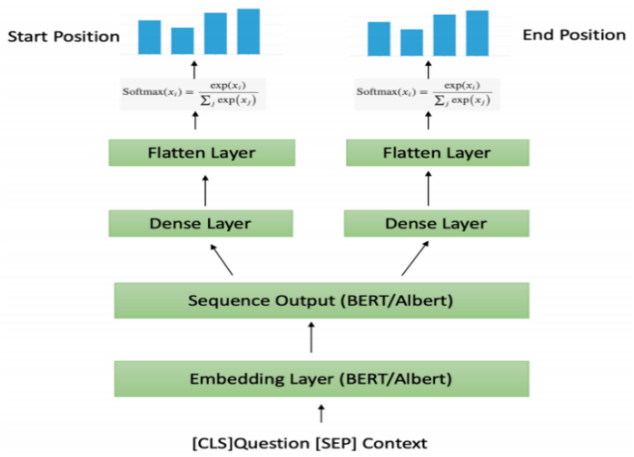

Fig. 6. BIRD-QA: BERT-style QA model development architecture.

Baseline Models: Pretrained BERT and ALBERT Models

We did not use the codes from the published work on fine-tuning BERT and ALBERT because they were not easy to be adjusted. Instead, we implemented our own code using TensorFlow 2, and HuggingFace Transformers Library [27] for tokenization. The implementation code will make available for reproducibility. We used the results of published paper on SQuAD dataset as our baseline which is reported in TABLE I.

Approach I: Feature-based Training (FeT) on SQuAD 1.1

Experiment: First, we applied a feature-based training (FeT) approach to extend pretrained BERT and ALBERT models. We froze the parameters of all 12 transformer encoder layers of BERT and ALBERT, and used them to generate static contextualized word embeddings. These embeddings are fixed and cannot be updated during additional training. Devlin et al. [10] concatenated the outputs of the last four hidden transformer layers and used it as static BERT embedding. We applied the same method for BERT. For ALBERT, we just used the output of the last transformer layer because ALBERT uses cross-layer parameter sharing, which makes it difficult to extract the output of each hidden transformer layer. Finally, we used a dense layer with softmax activation for classification. Fig. 6 shows the architecture of our model for SQuAD.

TABLE II. APPROACH I - FET: RESULTS FOR SQUAD 1.1 DEV SET

Model	*Strategy*	*EM*	*F1*
BERT-base + FeT	Exclusion	25.8	37.9
BERT-base (Concat Last Four Hidden) + FeT	Exclusion	32.7	44.2
ALBERT-base + FeT	Exclusion	**37.6**	**50.1**

Results: The results of Approach I, FeT, are shown in Table II. As shown in the table, most results were unsatisfactory. The extended ALBERT-base model gave the best result for the FeT approach. Our observation from the results were that the models were not able to capture important aspects of the questions and sequence from the data and failed to retrieve the correct answers.

TABLE III. APPROACH II - EFET: RESULTS OF SQUAD 1.1 DEV SET

Model	*Strategy*	*EM*	*F1*
BERT-base + EFeT			
+BiLSTM	Exclusion	62.3	71.8
+CNN	Exclusion	53.6	62.8
+CNN+LSTM	Exclusion	62.6	72.2
BERT-base (Concat Last Four Hidden) + EFeT			
+BiLSTM	Exclusion	66.2	74.7
+CNN	Exclusion	56.0	64.9
+CNN+BiLSTM	Exclusion	66.6	75.4
ALBERT-base + EFeT			
+BiLSTM	Exclusion	70.7	79.7
+CNN	Exclusion	63.6	73.3
+CNN+BiLSTM	Exclusion	**72.2**	**81.2**

Approach II: Enhanced Feature-based Training (EFeT) on SQuAD 1.1

Experiment: To improve the results of the FeT approach, we explored the feature-based training approach further by enhancing the architecture of the model (EFeT). As we froze the weights of the pre-trained model, it was difficult for only one dense layer to understand and extract information from sequence embeddings. To enable the model to learn and incorporate more features and sequence information in the embedding, we added a bidirectional Long Short Term Memory (BiLSTM) [26] before the classification layer. We also experimented with Convolutional Neural Network (CNN) and (CNN + BiLSTM).

Results: The results of Approach II, EFeT, are shown in Table III for the different variations of the enhanced models. While the different BERT variations give similar results, the CNN+BiLSTM variations are clearly able to capture more spatial and sequential features and give better results. In general, the recurrent neural network models i.e., the BiLSTM and the CNN+BiLSTM improve the outcome. ALBERT-base EFeT models outperform the BERT-base models in both FeT and EFeT approaches.

Approach III: Fine-Tuning based Training (FiTT) on SQuAD 1.1 and 2.0

Experiment: For fine-tuning approach, we used the same approach as shown in Fig. 6. SQuAD 1.1 is a smaller dataset and we could do more experiments with SQuAD 1.1 to choose the best model. After observing that ALBERT consistently performed better than BERT in FeT and EFeT approaches, we continued experimenting with both datasets for Approach III, FiTT. We ensured that all parameters of the pretrained model were trainable in this first Fine-Tuning approach. As BERT-style pretrained models are deep enough, adding more layers for fine-tuning on a downstream task may even get a lower performance. Therefore, we just used one additional dense layer after the embedding layer to train the BERT-style models on question answering tasks. To train on the SQuAD 2.0 dataset, we applied the two strategies namely the exclusion and the chunking strategies to handle long sequences and report the same in the results table. We chose the best model on SQuAD 1.1 and evaluated it on SQuAD 2.0.

TABLE IV. APPROACH III – FITT: RESULTS OF SQUAD 1.1 DEV SET

Model	*Strategy*	*EM*	*F1*
ALBERT-base + FiTT	Exclusion	**79.5**	**87.4**
BERT-base + FiTT	Exclusion	78.1	86.0

TABLE V. APPROACH III – FITT: RESULTS OF SQUAD 2.0 DEV SET

Model	*Strategy*	*EM*	*F1*
ALBERT-base + FiTT	Chunking	**75.4**	**78.8**
ALBERT-base + FiTT	Exclusion	74.5	77.8

Results: The results of Approach III, FiTT, are shown in Tables IV and V. As shown in Table IV, similar to previous two approaches, ALBERT-base pre-trained model gave a better result than BERT. Compared to FeT and EFeT from approaches I and II, FiTT achieved a significant improvement for BERT and ALBERT. Therefore, we took the best model ALBERT-base + FiTT, and then trained and evaluated the model on SQuAD 2.0. The results of SQuAD 2.0 are shown in Table V. We found that the chunking strategy achieved a better performance than the exclusion strategy.

Approach IV: Fine-Tuning based Training with Enhancement on Unanswerable Questions (FiTTE) on SQuAD 2.0

Experiment: This approach is only designed for SQuAD 2.0. As discussed in Section III.B, we labeled all unanswerable questions with (start position: 0, end position: 0), which are the positions of the "[CLS]" token. Whenever we got an output of "[CLS]", we knew that this question was impossible to be answered. Therefore, we could use the same model as shown in Fig. 6. However, in this approach, we force our

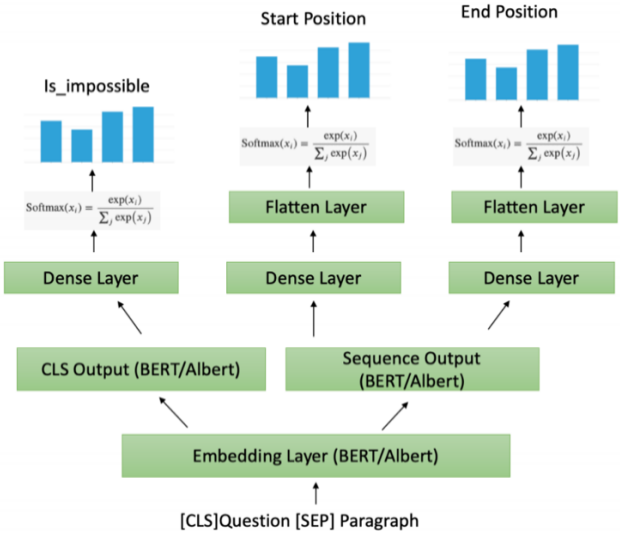

Fig. 7. Approach IV: FiTTE model

model to learn more about whether the question is answerable or not by computing one more cross entropy loss for feeding "is impossible" label as shown in Fig. 7. For this label, 1 indicates impossible to answer the question, and 0 otherwise. We experimented with both BERT-style CLS embedding and sequence embedding to classify this label.

TABLE VI. APPROACH IV – FITTE: RESULTS OF SQUAD 2.0 DEV SET

Model	*Strategy*	*EM*	*F1*
ALBERT-base + FiTTE	Chunking	**74.3**	**77.6**
ALBERT-base + FiTTE	Exclusion	74.0	77.5

Results: The results of Approach IV, FiTTE, are shown in Table V. ALBERT-base+FiTTE with chunking strategy achieves a better performance in this approach. However, the result is not as good as that from approach III.

C. Validation for Departmental Data

We used the Speech-To-Text (STT) function of IBM Watson API for enabling voice interaction with the QA service during the implementation and validation of the university QA real life use case scenario. We present a

Fig. 8. Screenshot of our QA system in conversation

screenshot from our BIRD-QA framework. An example question, "Which undergraduate program does school of computing offer?" was asked using voice. The Speech-To-Text (STT) API incorrectly translated the word "which" and "offer" as shown in Fig. 8. Our QA system still produced a correct answer.

D. Critical Discussion

We tried to improve the BERT and ALBERT-base models to develop a simpler and faster model for online real time QA and get better results for QA than what is reported in Table I. However, the original BERT and ALBERT-base codes were using TensorFlow 1 which was not easy to extend and did not report the results on SQuAD 2.0. Therefore, we report the performance of our versions of the extended base models. We tried to apply feature based training because it allows to store the static embedding. Therefore, we could generate and store static embeddings and use the same for evaluation and information retrieval for QA. As a result, for the online real time service, we would not need to use the entire BERT or ALBERT model to generate the embeddings for each QA all the time. Thus static embedding could save a lot of computational power. By contrast, fine tuning approach generates dynamic embeddings which cannot be pre-computed and stored. However, our experiments showed that fine tuning approach for BERT and ALBERT got a better result. Our fine-tuning results for the BERT and ALBERT-base models are not as good as the published results but by using our own base models, we could experiment and evaluate a variety of approaches and models to select a good model. We believe that with more iterations of training, we can get similar results as the published results for the base models.

During implementation, the most challenging part was the evaluation. In the case of using EM score and F1 score on text data, we needed to ensure that some specific cases such as unexpected symbols, and unexpected blanks would not influence the final results. Our observation from applying both feature-based and fine-tuning based approaches to SQuAD 1.1 was that a fine-tuning approach were better than a feature-based approach for BERT and ALBERT. Fine-tuning on pretrained ALBERT achieved the best performance with 79.5% EM score and 87.4% F1 score among all the models. Compared to the BERT model, ALBERT only contains 1/10 parameters, which greatly reduces the resource demands for the GPU and the RAM. Therefore, for future work, we can update ALBERT-base model to ALBERT-large, and even ALBERT-xlarge. For the same reason, we can apply fine-tuning approach to ALBERT on SQuAD 2.0. We experimented with two strategies that are described in Section III.B for longer sequences. Table IV shows that Approach III (ALBERT-base + FiTT) with strategy 1 achieves a better performance. It gets a 75.4% EM score and 78.8% F1 score. We also evaluate our model using departmental data which is shown in Section C.

We did not use the codes from published work on fine-tuning BERT and ALBERT because the code was not easy to modify for the experiments. Our results for the BERT and ALBERT-base models were not as good as the published results to start with. Therefore, we report the performance of our versions of the base models and their extensions. With more iterations of training, we believe that we could get similar results as the published results of the base models.

V. Conclusion

With the rise in big data and online cloud-based services, question answering systems are becoming critical to provide satisfactory customer services for business purposes. The traditional rule-based systems are difficult to build for all possible customer questions for big data. The advancements in NLP and deep learning techniques can enable QA systems to address the shortcomings of the rule-based systems and automatically extract or generate responses where question-answer pairs are not available in the knowledge base. In this paper, we present our study on developing an information retrieval-based QA system and validate its performance using benchmark SQuAD dataset. We applied two strategies to process long sequence of text input. We applied four different feature and fine-tuning based approaches to implement and validate a variety of extended BERT and ALBERT-base models to improve the embedding for the QA task. Our validation results show an improvement of the extended BERT and ALBERT models over their respective base models for the QA task. Our BIRD-QA models can not only answer the question that is possible to answer but can also detect if the question cannot be answered, and thereby, refuse to answer these questions. Due to the superior performance and less resource intensive architecture, we used the pretrained ALBERT model and then fine-tuned it on SQuAD 2.0 dataset for domain specific QA. We built our own knowledge base using the departmental webpages of a university to demonstrate a use case scenario of a question answering service.

Since our QA system has a limited input size for the referenced context, it is impossible to efficiently match the query with the complete sentences in our knowledge base. As we discussed in Section III.B, for future work, we want to split the unstructured data into multiple chunks and then search the answer in each chunk one by one. However, this can be very slow if we have a large knowledge base. Therefore, we would like to explore a ranking algorithm to find the most relevant context to retrieve the answer more efficiently. We also plan to extend our KB and apply larger base models for improved performance.

Acknowledgment

The research was done in collaboration with IBM and was funded by NSERC Canada grants.

References

[1] Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. Superagent: A customer service chatbot for e-commerce websites. In Proceedings of ACL 2017, System Demonstrations, pages 97–102, 2017

[2] Kyo-Joong Oh, Dongkun Lee, Byungsoo Ko, and Ho-Jin Choi. A chatbot for psychiatric counseling in mental healthcare service based on emotional dialogue analysis and sentence generation. In 2017 18th IEEE international conference on mobile data management (MDM), pages 371–375. IEEE, 2017.

[3] Kulothunkan Palasundram, Nurfadhlina Mohd Sharef, Nurul Nasharuddin, Khairul Kasmiran, and Azreen Azman. Sequence to sequence model performance for education chatbot. International Journal of Emerging Technologies in Learning (iJET), 14(24):56–68, 2019.

[4] Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. AliMe chat: A

sequence to sequence and rerank based chatbot engine. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 498–503, Vancouver, Canada, July 2017. Association for Computational Linguistics

[5] Jurgita Kapo˘ci¯ut˙e-Dzikien˙e. A domain-specific generative chatbot trained from little data. Applied Sciences, 10(7):2221, 2020.

[6] Adam S. Miner, L. Laranjo, and A. Kocaballi. Chatbots in the fight against the covid-19 pandemic. NPJ Digital Medicine, 3, 2020.

[7] Maali Mnasri. Recent advances in conversational nlp: Towards the standardization of chatbot building. arXiv preprint arXiv:1903.09025, 2019.

[8] Panitan Muangkammuen, Narong Intiruk, and K. Saikaew. Automated thaifaq chatbot using rnn-lstm. 2018 22nd International Computer Science and Engineering Conference (ICSEC), pages 1–4, 2018.

[9] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. arXiv preprint arXiv:2007.05558, 2020.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[11] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv, abs/1909.11942, 2020.

[12] Gobinda G Chowdhury. Natural language processing. Annual review of information science and technology, 37(1):51–89, 2003.

[13] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36–45, 1966.

[14] Amber Nigam, Prashik Sahare, and Kushagra Pandya. Intent detection and slots prompt in a closed-domain chatbot. In 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pages 340–343. IEEE, 2019.

[15] Nazakat Ali. Chatbot: A conversational agent employed with named entity recognition model using artificial neural network. arXiv preprint arXiv:2007.04248, 2020.

[16] Kennedy Ralston, Yuhao Chen, Haruna Isah, and Farhana Zulkernine. A voice interactive multilingual student support system using ibm watson. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pages 1924–1929, 2019.

[17] Richard Csaky. Deep learning based chatbot models. arXiv preprint arXiv:1908.08835, 2019.

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[19] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. arXiv preprint arXiv:1804.09541, 2018.

[20] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

[21] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016.

[22] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. arXiv preprint arXiv:1806.03822, 2018.

[23] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[24] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226, 2018.

[25] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2018

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9:1735–1780, 1997

[27] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R´emi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. In In Advances in Neural Information Processing Systems, 2017

[29] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[30] Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. arXiv preprint arXiv:1610.07149, 2016.

[31] Sumikawa, Yasunobu, et al. "Supporting creation of FAQ dataset for E-learning chatbot." *Intelligent Decision Technologies 2019*. Springer, Singapore, 2020. 3-13.