Summer 2019

EagleBot: A Chatbot Based Multi-Tier Question Answering System for Retrieving Answers From Heterogeneous Sources Using BERT

Muhammad Rana

EAGLEBOT: A CHATBOT BASED MULTI-TIER QUESTION ANSWERING SYSTEM FOR RETRIEVING ANSWERS FROM HETEROGENEOUS SOURCES USING BERT

by

MUHAMMAD RANA

(Under the Direction of Mehdi Allahyari)

ABSTRACT

This paper proposes to tackle Question Answering on a specific domain by developing a multi-tier system using three different types of data storage for storing answers. For testing our system on the University domain we have used extracted data from the Georgia Southern University website. For the task of faster retrieval we have divided our answer data sources into three distinct types and utilized Dialogflow's Natural Language Understanding engine for route selection. We compared different word and sentence embedding techniques for making a semantic question search engine and BERT sentence embedding gave us the best result. For extracting answers from a large collection of documents we also achieved the highest accuracy using the BERT-base model. Besides trying with the BERT-base model we also have achieved competitive accuracy by using BERT embedding on paragraph split documents. We have also been able to accelerate the answer retrieval time by a huge percentage using pre-stored embedding.

INDEX WORDS: Chatbot, Question answering system, BERT, Multi-tier Q A system

EAGLEBOT: A CHATBOT BASED MULTI-TIER QUESTION ANSWERING SYSTEM FOR RETRIEVING ANSWERS FROM HETEROGENEOUS SOURCES USING BERT

by

MUHAMMAD RANA

B.S., Bangladesh University of Engineering and Technology, Bangladesh, 2016

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

EAGLEBOT: A CHATBOT BASED MULTI-TIER QUESTION ANSWERING SYSTEM FOR RETRIEVING ANSWERS FROM HETEROGENEOUS SOURCES USING BERT

by

MUHAMMAD RANA

Major Professor:	Mehdi Allahyari
Committee:	Pradipta De
	Andrew Allen

Electronic Version Approved:
July 2019

DEDICATION

I dedicate this thesis and all my academic achievements to my parents, my wife, my family, my honorable teachers and all my friends who inspired me to cross all the hurdles in my life. I am grateful to these people for everything in my life.

ACKNOWLEDGMENTS

I would like to sincerely thank Dr. Mehdi Allahyari for his guidance throughout these years and for being such a great friend and mentor. I would also like to thank Dr. Pradipta De, Dr. Andrew Allen and Dr. Muralidhar Medidi for their assistance throughout the project. I can never thank enough Scott for all his supports on the technical issues. The acknowledgement won't be complete without mentioning Amy Smith, Janice Stanford & Brittni Favorite and their contribution throughout the Data Collection phase.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

One of the big challenges at every university or college is how to answer the vast amount of students' questions in a fast and efficient manner. Students usually check the university website, send an email, make phone calls or meet in person with the appropriate staff to get their information. Although these methods work, they are limited in certain ways: (a) since the number of university employees are restricted; (b) university staff are not available around the clock and more importantly (c) students mostly require to carry out extra work such as navigating the university website to extract the desired information from there. The same challenge exists in many other domains where we need constant human assistantship to tackle user queries. So, we propose a chatbot approach, which we have named Eaglebot, to answer the variety of questions that students usually ask on the university domain. For the case study, we use the Georgia Southern University (GSU) website.

A chatbot or an artificially intelligent tool is a conversation agent that interacts with users via auditory or textual methods using natural language processing system. In recent years, Chatbot technology has been grown. Many chatbots across a wide range of domains such as customer service, e-commerce, healthcare, education and training have been developed and deployed. Most chatbots are accessed through virtual assistants such as, Apple Siri, Google Assistant or Amazon Alexa, or via messaging apps such as Facebook Messenger or WeChat. Of various usage categories education is an area where chatbot technologies has shown a great deal of potential in terms of enhancing student engagement and learning. Course assessment, efficient teaching assignments, intelligent tutoring systems, and improving student engagement are some challenges that are faced by educational chatbots these days. Although chatbots are getting prevalent, they usually fail to deliver a seamless and delightful user experience due to several reasons including lack of context,

unclear intentions, meaningless responses or inadequate usability.

The majority of traditional chatbots are rule-based. Where a user asks a question a traditional chatbot tries to find the answer by using some pre-defined rules. Though there are few chatbots which are not rule based, they are limited to their scope. Most of these chatbots try to retrieve an answer using a large pre-stored question-answer storage or from previous discussion threads. In our knowledge there is no such current chatbot system where the system can tackle all types of probable questions asked in a domain. This was our motivation to work on this domain. We classified the probable questions into 3 major categories. For handling these three categorical questions, we proposed a system architecture using different machine learning and non-machine learning based techniques. For understanding user's intention from a question, we used Dialogflow's Natural Language Understanding (NLU) toolkit. We tried several sentence similarity finding models to find similarity between user's query and expected answer. Then, for extracting an answer span from a huge collection of documents, we used Google's recent groundbreaking publication BERT model. BERT (Devlin et al., 2018), the latest refinement of a series of neural models has led to impressive gains in many natural language processing tasks, ranging from sentence classification to question answering to sequence labeling. Finally, we deployed EagleBot as a chatbot on two different platforms so that the users can interact with this both from laptops and from mobile phones with or without using Facebook.

CHAPTER 2

RELATED WORKS

A chatbot or an artificially intelligent tool is a conversation agent that interacts with users using a natural language that simulates human conversations (Shawar and Atwell, 2007). Chatbot technology has been growing rapidly in recent years. Many chatbots have been created and deployed across a wide range of domains such as customer service, e-commerce, healthcare, education and training. Apple Siri, Google Assistant and Microsoft Cortana are at the forefront of this technology. Education is an area where chatbot technologies have demonstrated a great deal of potential in terms of enhancing student engagement and learning. Course assessment, efficient teaching assignments, intelligent tutoring systems, and improving student engagement are among the challenges in education that educational chatbots have been trying to address recently (Holotescu, 2016) (Fadhil and Villafiorita, 2017).

A recent study revealed four main motivation for using chatbots: (a) productivity, i.e. chatbots are fast, easy and convenient to use to obtain information; (b) entertainment, i.e. chatbots are fun and entertaining to use; (c) social/relational, i.e. chatbots can augment interactions between humans, for example, to avoid loneliness or fulfill an eagerness for socialization; and (d) novelty/curiosity, i.e. chatbots are novel and explore the limits or their abilities (Brandtzaeg and Følstad, 2018). Of these four categories, productivity with 48% was the most frequent motivational factor of the respondents, which indicates chatbots help users obtain information or assistance in an efficient and timely manner. These results highlight the importance of integrating AI in different disciplines. Education in particular can considerably benefit from chatbot technology.

Chatbots are increasingly gaining popularity in the education domain especially in the university domain for different tasks.

In one study, an android application was presented containing an educational chat-

bot designed for visually impaired people (Kumar et al., 2016). The application can be launched with Google Voice Search and is used by asking questions in spoken natural language. The application will then convert it to text and run the query against the AIML database or, if no pre-saved answer was found, against the Wikipedia API. The answer is then returned to the application as both text and voice. Though the application of this chatbot is noble, the retrieval architecture is a simple rule-based one.

Another study presented a chatbot design which was implemented to answer FAQs regarding university related questions in an efficient and accurate way (Ranoliya et al., 2017). This chatbot was based on AIML and the researchers had the goal to further improve the capabilities by eventually also using Latent Semantic Analysis.

Jill Watson, the Georgia Tech teaching assistant chatbot demonstrated the strong viability of chatbots in the educational domain (Goel and Polepeddi, 2016). Jill Watson showed promise as an alternative to teachers in the near future. Georgia Tech Computer Science Proffessor Dr. Ashok Goel built this chatbot to help students on their assignment related questions in one of his Artificial Intelligence courses. JW1 (Jill Watson version 1) was built using IBM Watson APIs. JW1 had a memory of question answer pairs from previous semesters organized into categories of questions.

In 2016, a Boston based EdTech startup named Admithub gained huge success by launching a chatbot called *Pounce* at Georgia State University for reducing Summer Melting and helping high school students on their transition period to college (Page and Gehlbach, 2017). Their chatbot based enrollment approach gained huge success. Admithub claimed that, they had been able to reduce the summer melting rate by 24% and to increase the enrollment rate by 3.9% during that year.

After Admithub's success on Georgia State University, they introduced a similar chatbot at Georgia Southern University called *GusBot* for the same purpose, increasing the enrollment rate in Georgia Southern University. In both of these above mentioned chat-

bots, Admithub used a fixed knowledge base for retrieving answers. That means, they had a fixed set of questions and they tried to match any new question with that set of questions.

Besides these two applications, chatbots were deployed in university domains for answering questions for a specific course by using previous years' chat discussion board as training data (Feng et al., 2006). All these aforementioned chatbots systems work for a very specific domain and don't deal with the whole university domain. As of now, as to the best of our knowledge, there is no such integrated system for answering all types of questions asked in the university domain. Our experiment shows that it's possible to build a chatbot for handling the whole domain using our proposed architecture.

CHAPTER 3

SYSTEM DESIGN

Figure 3.1 gives a high level overview of the proposed architecture to retrieve answers of a question asked on a specific domain in the shortest possible time. This section includes a breakdown of the server and client implementations.

3.1 SERVER SYSTEM

In our project, we have built a three-tier system architecture to tackle three different groups of questions asked in a specific domain. We classified the questions into three different groups as the answers to these questions were retrieved from three distinct sources of data storage. We classified them as: I) QA on Structured Data, II) QA on FAQ Data and III) QA on Unstructured Passage Data. For example: retrieving the name of the courses taught by a specific teacher from a course table can be considered as type I (finding answers for this type of questions are convenient by using a structured tabular form), retrieving answers from a frequently asked question list can be considered as type II and retrieving answers from any document in the web within that domain can be considered as type III. We use Dialogflow's NLU engine to understand the user's query and identify the entities and the intent of the question, which helps us to select the route for finding the answer. For answering type II (FAQ module), both TF-IDF and several Sentence Embedding based models like Infersent (Hassan et al., 2019) (released by Facebook at 2017), Universal Sentence Embedding (USE) and BERT (Devlin et al., 2018) (both released by Google respectively at 2017 and 2018) have been used to compare the retrieval accuracy and run-time. Then, for answering type III queries, BERT-base model has been used to distinguish the performance with our baseline method BiLSTM. Additionally, we have tried with unsupervised approaches using BERT embedding to compare with our baseline BiLSTM model.

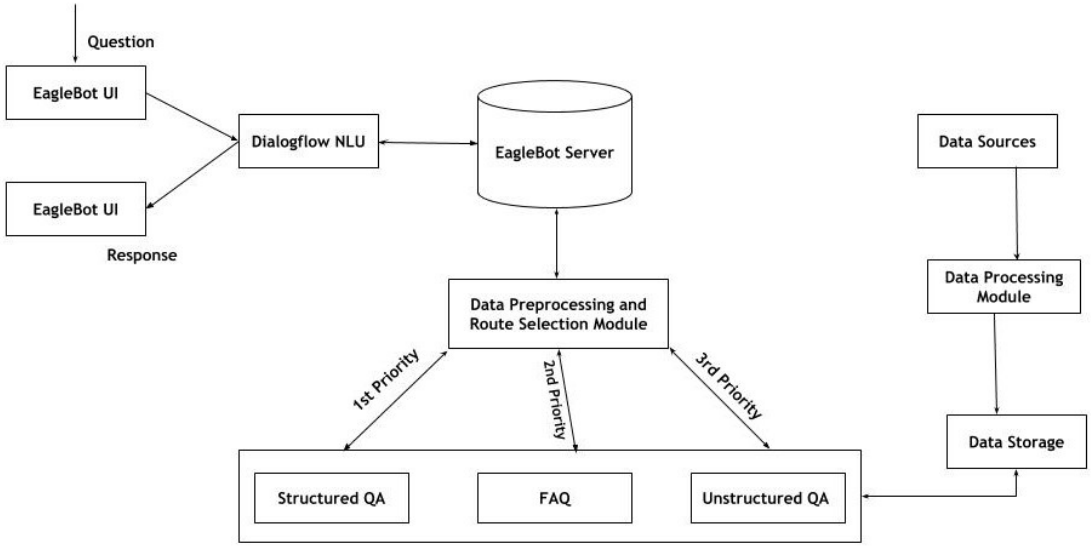

Figure 3.1: High Level Architecture of EagleBot

3.1.1 Type I: Answering Question from Structured Tabular Data

Dialogflow

Dialogflow is a conversational agent building platform from Google. It is a web-based platform that is accessible from any web browser. We have used Dialogflow to capture the intent of a user's query, entity (the most important pieces of tokens in a text) within the query and context of a query during a multi-turn conversation.

Intent

Any basic flow of chatbot conversation involves three steps; I) Inputting user query, II) Parsing the user query and III) Returning response to that query. Intents do the task of mapping the user's input to response. In each intent, we defined examples of the user's utterance, what to extract from that utterance and how to respond to that. For example, phrases like *"I want pizza ," "Get a pizza,"* or *"Order pizza"* all means they are indicating

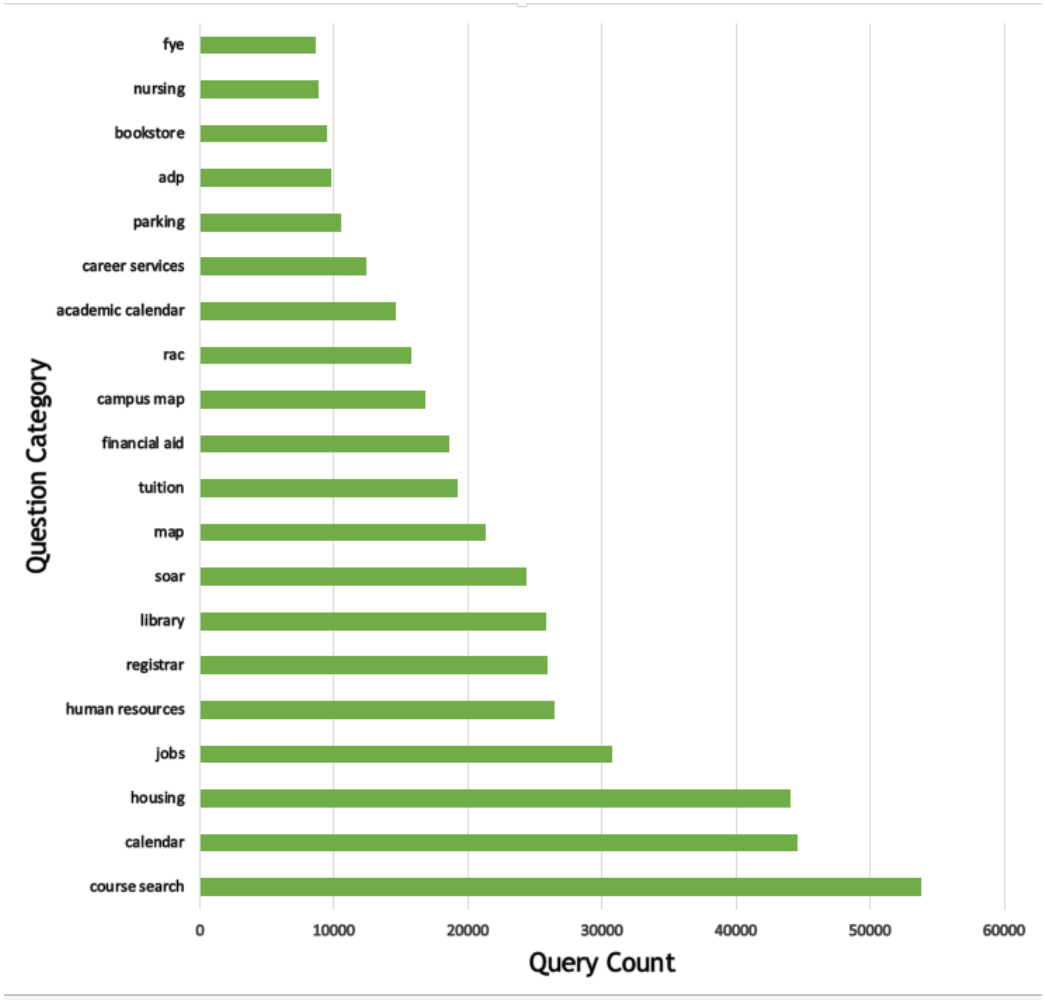

Figure 3.2: Georgia Southern Historical Search Data

the same intent ***Pizza Ordering Intent***.

Entity

Entities are Dialogflow's mechanism for identifying and extracting useful data from natural language inputs. While intents allow any chatbot agent to understand the motivation behind a particular user's input, entities are used to pick out specific pieces of information that the user have mentioned. For example, if a user's input phrase is *"Please order a **12** pizza,"* dialogflow match ***"12"*** as ***Pizza Size*** entity.

Context

Contexts represent the current state of a user's request and allow a chatbot agent to carry information from one intent to another. In our pizza example, if a user asks: *"Order me a pizza"*, the chatbot needs to know more details about the order like: pizza size, toppings, extra sauce and few more specifications. For collecting all these information on a multi-turn conversation and staying on the same phase we needed contexts.

The whole answer retrieval process of type I will be clarified by the example below.

Question: Is **Dr. X** teaching **CSE 101?**

Intent: "Course and Teacher Related Search"

Entity: "teacher_name": "Dr. X", "course_id": "CSE 101"

For example, from the above question our system maps the question as a type I question with the intent type of *Course and Teacher Related Search* and extracts two entity *teacher_name* and *course_id*. Then, flask powered backend converts this question to a MongoDB search query like below to extract the desired result from the database, where we pre-stored many frequently-searched tables.

collection.find("Title": "$regex": ".*" + course_id + ".*", "teacher_name": "$regex":

".*" + teacher_name + ".*","Title": 1, "teacher_name ": 1, "course_name": 1, "_id": 0)

Finally, from the returned query result, we create a rich response message to show the result to the user. And, if the question doesnt match with Type I, it automatically proceeds to Type II and III.

In order to test EagleBot's performance on type I data, we first needed to design multiple intents. For selecting the topics, we relied on Georgia Southern historical search data as shown in figure 3.2. From the list, we arbitrarily selected 3 topics (Course Search, RAC timing & Event Calendar) for testing our type I QA. From these three topics, we created several probable intents in Dialogflow. For example, from the course search topic, we created multiple intents such as: course teacher search, course timing search, course availability search, etc. Finally, for all the intents that we created, we trained them by feeding multiple probable sentences for each of the intents. For example, for training the ***course teacher search*** intent, we used phrases such as "***who is teaching course X?***", "***who is teacher in course X?***", "***hey bot, do you know who is teaching course X this time?***" and many more. The downside of this module is that, it is a hard coded and manual technique.

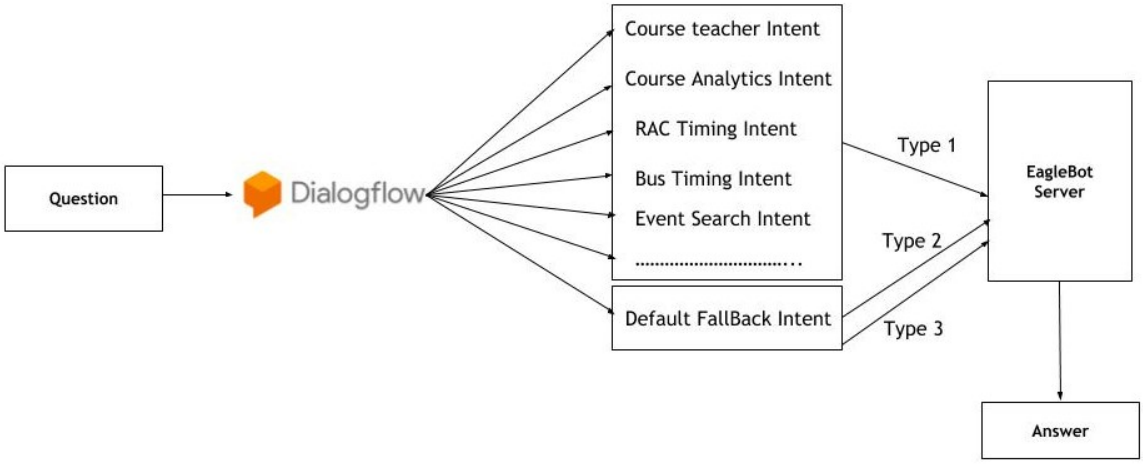

Figure 3.3: Route Selection For EagleBot Using Dialogflow

3.1.2 TYPE II: ANSWERING QUESTION FROM EAGLEBOT FAQ KB

If the question does not match with Type I, our system pursues the answer from our Frequently Asked Question Knowledge Base (FAQ KB). Dialogflow's Default Fallback Intent was used to select the route as shown in figure 3.3. In this module, our system tries to extract the answer using TFIDF with cosine similarity and three Sentence Embedding based models: Infersent, Universal Sentence Embedding (USE), and BERT model. For testing the EagleBot on type II questions, we extracted about 400+ FAQs from Georgia Southern's several FAQ web pages and saved them into a .csv (comma separated value) file. With the help of the above mentioned four models, we converted all the FAQ questions into fixed-size vectors. Finally, by using cosine similarity technique, we found which one is the closest question to the query.

TFIDF

Term Frequency-Inverse Document Frequency (TF-IDF) gives the idea of how important a word is to a document (Ramos et al., 2003). TF-IDF and cosine similarity find out the closest question with an answer from our FAQ KB. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contains the word. It helps to adjust the fact that some words appear more frequently.

Sentence Embedding

Sentence embedding encodes sentences in fixed-length dense vectors. An effective vector representation is the cornerstone for many real-world NLP/AI applications. A common approach for calculating sentence embedding is to compute the Bag-of-Words (BoW) of the word vectors and then using the arithmetic mean of the word embeddings. Although

this method provided us with limited performance, a recent method named Smooth Inverse Frequency (SIF) has demonstrated notable improvements over traditional weight averaging by using weighted averages and modifying them using Singular-Value-Decomposition (SVD) (Perone et al., 2018). However, the recent works on Sentence Embedding using supervised learning has outperformed all the previous works on this domain. Below is a brief description of the three most recent works in this problem domain.

Infersent

Inferesent is a sentence embedding method that provides semantic representations for English sentences (Conneau et al., 2017). It uses Stanford Natural Language Inference (SNLI) data corpus (a set of of 570k pairs of sentences labelled with 3 categories: neutral, contradiction and entailment) to train a classifier on top of a sentence encoder. This model is generalized for many NLP tasks. We used it for encoding our FAQ KB and to answer type II questions. We used Facebook's open source code for Infersent model from github.

Universal Sentence Encoder

Universal Sentence Encoder (USE) is one of the best universal sentence embedding models released by Google in 2018 (Cer et al., 2018). USE encodes any text into 512-dimensional embedding that can be used for a variety of NLP tasks. This model is based on Deep Averaging Network (DAN) which takes comparatively less time and memory for computation than the later one released by Google, called BERT. In order to use this model, we relied on Google's original implementation of USE.

BERT

BERT is a state of the art technique for measuring sentence embedding along with many NLP tasks (Devlin et al., 2018). It was released by Google at the end of 2018. Just

like Universal Sentence Encoder it also gives 512- dimensional embedding of any text. But during building this model, Google concentrated more on the accuracy of the task. So, memory and time-wise, this model is not as efficient as USE. Using Google's original implementation of BERT, the sentence embedding/vectorization task is a pretty complex one. For this purpose, we used Han Xiao's (Xiao, 2018) BERT-as-service library to map variable-length sentences into fixed-length vectors. The advantages of using this library is four fold. This library is really easy-to-use, fast (900 sentences/s on a single Tesla M40 24GB), scalable, and reliable (tested on multi-billion sentences).

3.1.3 TYPE III: ANSWERING QUESTION FROM UNSTRUCTURED PASSAGES

If EagleBot does not find a Type II answer with a certain confidence level, it indicates that we need to dive into the final module. Our final module is inspired by the recent success of DrQA (Chen et al., 2017) and BERTSerini (Yang et al., 2019). the workflow can be divided into two main parts: *1) Candidate Document Retrieval* and *2) Answer Span Retrieval*.

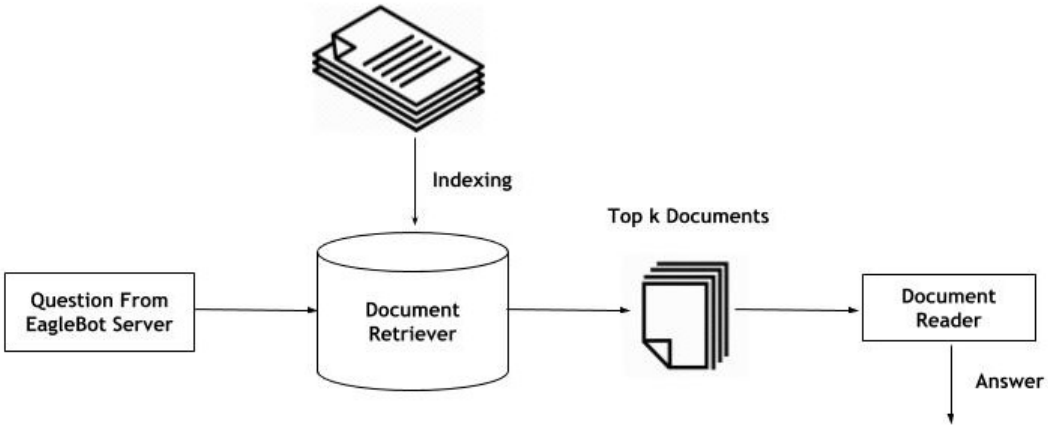

Figure 3.4: High Level Architecture of Type III Answer Retrieval Module

Candidate Document Retrieval

The candidate document retrieval task can be divided into two sub-tasks again: I) Indexing of all the documents and II) Retrieval of top *k* documents.

For testing this module, we used all the web pages from Georgia Southern University. For scraping all the 28,000+ web pages, we first used the sitemap.xml file to extract all the links. Then, from all the links, we extracted all the raw texts from the main container portion of the web pages. We ignored all the side bars, buttons, images, and navbars. We only focused on the texts from the middle container. We noticed that in most of the cases, the main portion of the texts come from the main container. Finally, we indexed all these documents using ElasticSearch. For our experimental purpose, we also compared the results retrieved from ElasticSearch against GoogleSearch.

Briefly, when a user's query comes to this module, the system fetches the top k=1 to 10 documents as the most probable candidate documents. Next, these candidate documents are appended together and ultimately fed along with the user's query into the deep learning models for predicting the most probable answer span.

Answer Span Retrieval

For answer span prediction, we used BiLSTM model (Chen et al., 2017) as our baseline model. But, our system gave better performance by using BERT-base (Devlin et al., 2018). After some fine-tuning using BERT-base and some linguistic updates, our system gave us the best result.

BiLSTM: One big advantage of using a BilSTM model over a lstm model is that the model reads an input context from both forward and backward (as shown in figure 3.5). So, the model gets a better understanding of the context using a BiLSTM model. Even human brains sometimes read in a bi-directional fashion when we read anything sarcastic or

very complex. Our baseline BiLSTM model for machine reading is inspired by the recent success of DrQA, where their document reader model showed impressive performance on machine comprehension of text (identifying answer span of a question from a list of selected documents). Given a question *q* consisting of *l* tokens *q1, . . . , ql* and a document or a small set of documents of *n* paragraphs where a single paragraph *p* consists of *m* tokens *p1, . . . , pm*, the system uses the BiLSTM model to predict the span of tokens that are most likely the correct answer (Chen et al., 2017). First, the model uses the paragraph to construct a feature vector using word embedding and few more feature extraction techniques i.e. exact match, token features and aligned question embedding. The system then feeds the feature vector constructed from the paragraph to a multi-layer BiLSTM model to create the paragraph encoding. The same technique is applied for constructing the question encoding. Finally, the system trains two different classifiers using the paragraph encoding and question encoding to predict two ends of the answer span.

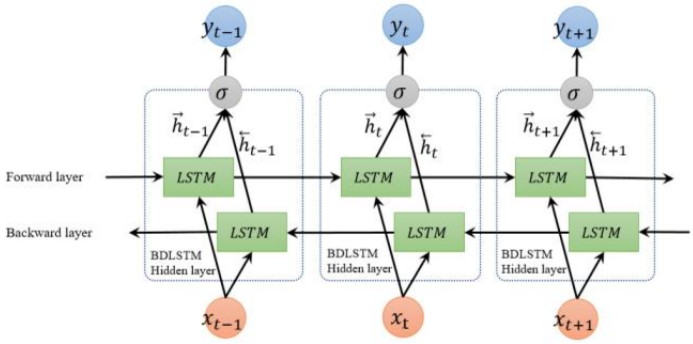

Figure 3.5: High Level Architecture of a BiLSTM model

BERT: BERT, which stands for Bidirectional Encoder Representation from Transformers, is a method of pre-training language representations. The key modules of BERT are two pre-training tasks. The first task Masked LM (MLM) aimed to break the limitation of the traditional unidirectional models and leverage the power of bidirectional models. They mask out 15% of the words in the input and run the entire sequence through a deep

bidirectional transformer encoder and then predict the masked words. The second task tries to learn the relationships between sentences. Given two sentences A and B, the model is trained to learn, if B is the actual next sentence that comes after A, or just a random sentence from the corpus.

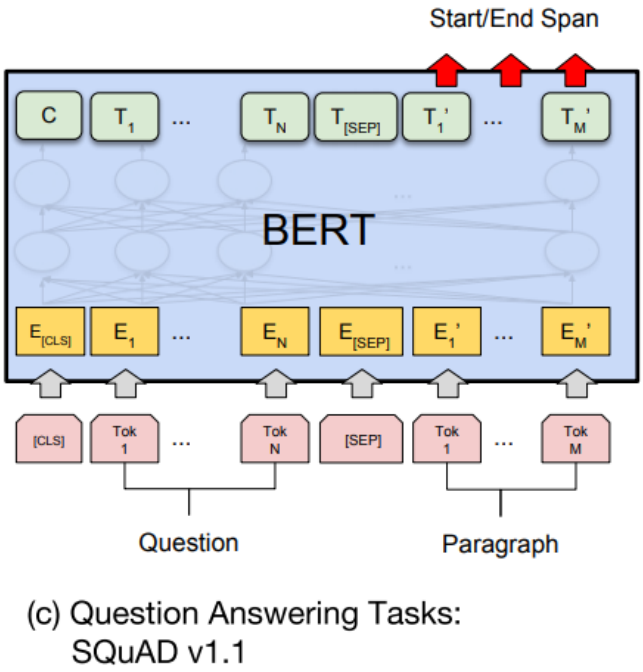

Figure 3.6: High Level Architecture of BERT Question Answering Model

BERT Fine-Tuning: Based on the original implementation of the multi-layer bidirectional transformer, BERT provides a few different model sizes. Considering our computational capacity, we choose BERT-base, Uncased model. We fine-tuned the BERT model using SQuAD 1.1 (Stanford Question Answering) (Rajpurkar et al., 2016) dataset. We further fine-tuned the saved BERT model using our custom SQuAD like university domain Question Answering dataset to feed more specific domain knowledge into the BERT model.

Answer Retrieval With Unsupervised Approaches Using BERT: We were also interested to see, how an unsupervised model was going to perform on this complex task. For experimenting this, we decided to chunk our candidate documents into sentences and paragraphs. Then, we embedded all the sentences and paragraphs as well as the question using bert-as-service as mentioned in the section 4.1.2.. And, finally we predicted the closest sentence/paragraph using cosine similarity and returned that sentence/paragraph. To our surprise, this technique achieved notable accuracy. Precisely, paragraph embedding gave us the better result than the sentence embedding. The probable reasoning is described in the results and discussions chapter.

3.2 Client System

For ensuring maximum number of user accessibility, we have deployed EagleBot in two separate platforms: I) Facebook and II) Web User Interface using Flask. Here is a brief description about the integration of these two frameworks with our EagleBot core server.

3.2.1 Facebook Messenger UI

Dialogflow supports integration with multiple popular platforms, such as Google Assistant, Facebook Messenger, Twitter, Slack, Telegram and many more. One study showed that 68% of U.S. adults use Facebook either online or on their cellphone (Smith and Anderson, 2018). Another study claimed that during first quarter of 2019, Facebook had 2.38 billion monthly active users (Clement, 2019). Considering this huge user base, we have deployed EagleBot in the Facebook Messenger platform. For the integration task with Facebook Messenger, Dialogflow generates an api key, which we needed to share between Dialogflow and Facebook. The biggest problem about using the Facebook Messenger UI is the issue of getting approval for publishing an app into Facebook Messenger platform. Considering the approval issue, we are currently using it as a test app where we can add up

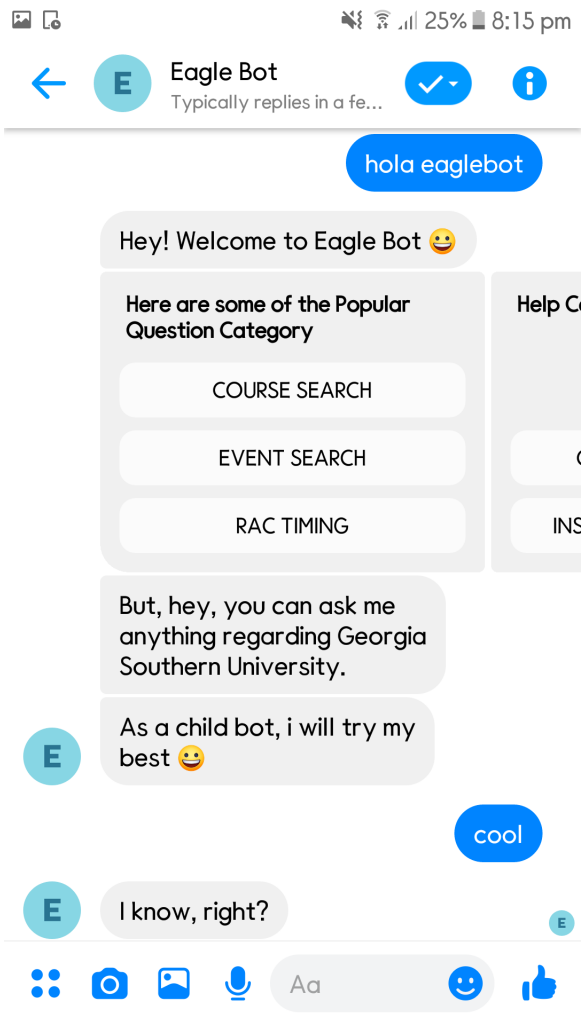

Figure 3.7: EagleBot Messenger UI

to 2000 testers.

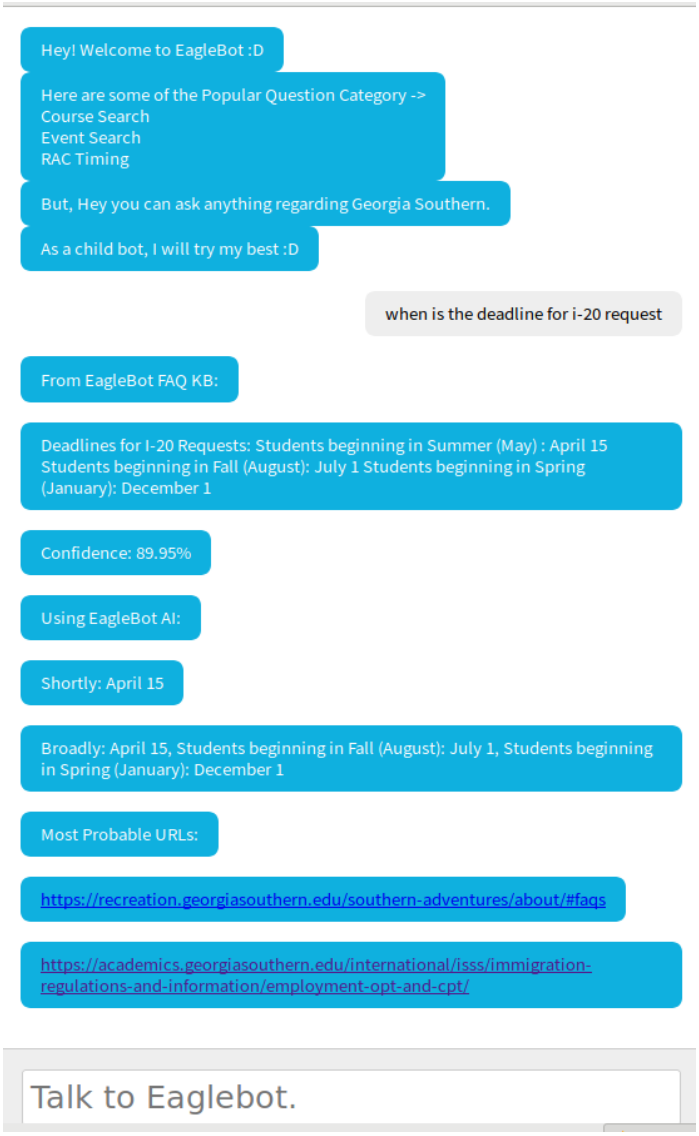

Figure 3.8: EagleBot Messenger UI

3.2.2 Web UI using Flask

Primarily there are two reasons for publishing EagleBot as an independent web UI:

I) Considering a portion of our user base, who does not have an account on Facebook.

II) Dialogflow has a default response timeout of 5 seconds. If the response does not come

back from EagleBot core server within 5 seconds, the messenger UI of EagleBot shows default timeout response, which is not desirable in a practical case.

In order to solve the above two issues, we built this web UI by using asynchronous call so that we can ensure the response back with any amount of waiting time. We used Flask for building this app. Besides, anyone with an internet access can use EagleBot through this web UI.

Question	**Context**	**EagleBot Response**
Who teaches CSCI 1302?	Structured Tabular Data (Type I)	The gym is open today, Friday (4/26/2019) from 6am to 9pm
Is gym open today?	Structured Tabular Data (Type I)	Dr. Mehdi CSCI 1302 a TR (2:00p-3:15p)
What is the deadline for i20 doc request?	From Eagle FAQ KB (Type II)	Deadlines for I-20 Requests: Students beginning in Summer (May): April 15 Students beginning in Fall (August): July 1 Most Probable URL: https://academics.georgiasouthern.edu/international/isss/prospective-students/
Where to mail my visa documents?	Where to Mail your Documents: ***You can mail all I-20 (F-1 Visa) documents to Ms.Tanya Brakhage (Statesboro) or Ms. Sara Nobles (Armstrong)***: You can contact our office at international@georgiasouthern.edu (type III)	You can mail all I-20 (F-1 Visa) documents to Ms. Tanya Brakhage (Statesboro) or Ms. Sara Nobles (Armstrong) Confidence: 89% Most Probable URL: https://academics.georgiasouthern.edu/international/isss/prospective-students/applying-to-georgia-southern/

Where to contact for scholarship related issues?	Apply for admission, apply for scholarships, and get to know your admissions counselor. ***Questions related scholarships please email to scholarships@georgiasouthern.edu. Questions related scholarships please contact 912.478.5391***	From FAQ KB: Questions related scholarships please email to scholarships@georgiasouthern.edu Confidence: 90 From BERT: ***Email gradadmissions@georgiasouthern.edu (Wrong Answer)*** Most Probable URL: https://admissions.georgiasouthern.edu/scholarships/

Table 3.1: Sample questions and answers generated from EagleBot with the context and question type

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Experimental Setup

The development environment that we used for our experimental setup has the following configurations: Operating System: Ubuntu 18.04, Processor: Intel(R) Xeon(R) W-2123, CPU: 3.60GHz, System Memory: 64 GB DDR4, GPU: Two GeForce GTX 1080

4.2 Model Accuracy Comparison For Type I Answer (Tabular Data)

High accuracy of type 1 answers depends on the proper detection of type 1 questions. For detecting a type I question, we had to train Dialogflow with many probable question sentences. If the question asked by the user is not trained by Dialogflow or slightly differed from the trained questions, the probability of the question's intent being misidentified by Dialogflow will be high. But, if Dialogflow identifies a type I question, it is almost certain to get the correct answer.

4.3 Model Accuracy Comparison For Type II Answer (FAQ)

Initially, we started implementing the model for finding Type II answers (FAQ Answer) by using TF-IDF. In terms of model loading time and word vectorizing time TF-IDF outperforms most other existing models. But, in terms of accuracy, TF-IDF falls below the expectation compared to our other sentence similarity models. The scenario is clearly depicted on figure 4.1 and table 4.1, where it shows that in our FAQ answer retrieval task, TF-IDF performs very poorly with an accuracy of little over 30%, whereas Universal Sentence Encoder (USE) and BERT embedding models have achieved more than 70% accuracy. The reason behind this result deviation is very clear.

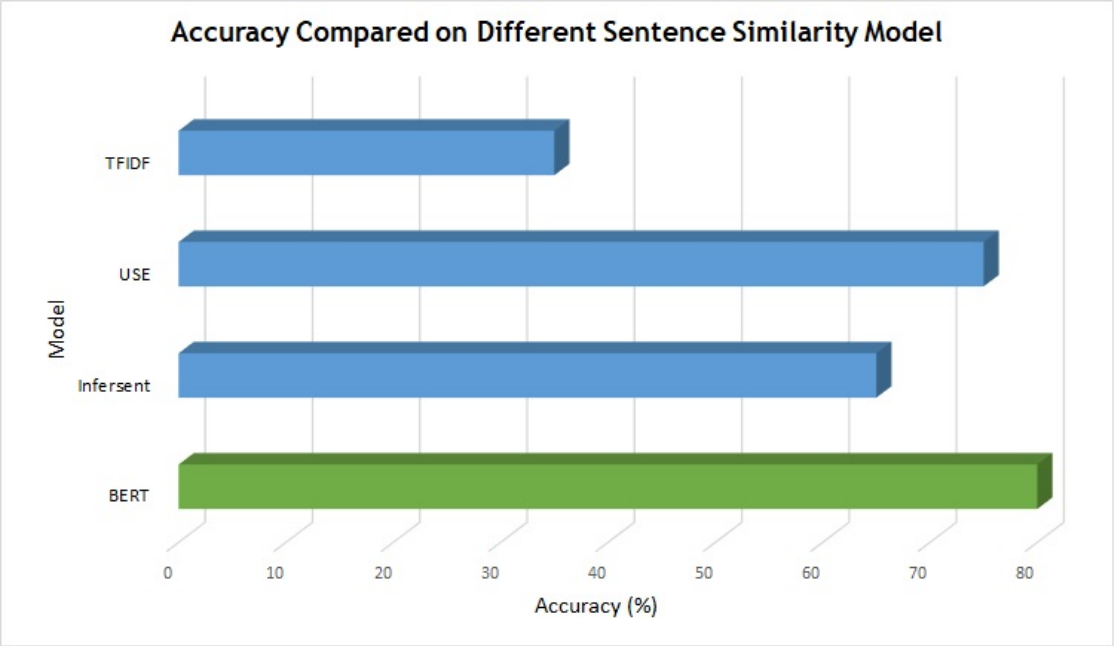

Figure 4.1: Accuracy Comparison on Type II Answer Retrieval Task

TF-IDF is a model based on Bag-of-Words (BoW), which means for finding a similar sentence it solely depends on the word frequency and it does not count the semantic similarity. That is why, when a user asks a question with alternative words, it confuses the correct answer with some other answers which have more word matching than the correct answer. For example, if a user asks ***"Where to mail visa papers?"***, the TF-IDF model will return ***"Where to mail visa fee?"*** with higher confidence than ***"Which address to send visa documents?"*** as the former question has more word matching than the later one though which is clearly not the desired answer.

Among the three other sentence similarity models, Infersent model by Facebook performed worse than the other two. Infersent can detect simple semantic relation relatively well but in case of little complex semantic relation, it fails to answer properly. For example, Infersent can catch that ***"where to send i20 documents"*** and ***"Which address to mail my visa/i20 documents?"*** both are similar sentences but it failed to detect the similarity be-

Model Name	**Accuracy (%)**
BERT Sentence Embedding	80
Infersent Sentence Embedding	65
USE	75
TFIDF	35

Table 4.1: Accuracy Comparison on Type II Answer Retrieval Task

tween ***"where to send visa papers"*** and ***"Which address to mail my visa/i20 documents?"***.

Both the Universal Sentence Encoder (USE) and BERT models have performed exceptionally well on finding similarities between sentences. Both the models were able to find complex semantic relations between sentences with a high confidence value. One example to mention, both of the models identified that ***"where to live in Georgia Southern"*** is the closest one to the sentence ***"Where can I find off-campus housing in Statesboro"*** from a list of 400+ sentences including a few very confusing ones. Though both of these models performed really well on the task, we chose BERT embedding as our final choice for Type II answer retrieval task for the reasons given below: I) BERT is the most recent publication from Google for this task and is an update over USE, so BERT has a relatively higher accuracy for this task and, II) Though both of the models have been successful on finding complex relations, BERT returned the answers with higher confidence than USE.

4.4 Embedding/Vectorization Time Comparison

For this experiment, we have used 400+ FAQ questions and recorded the time needed to vectorize/embed them. It is clear from figure 4.2 and table 4.2 that scikit-learn's TF-IDF implementation took the least amount of time to embed all the 400+ questions. Then the

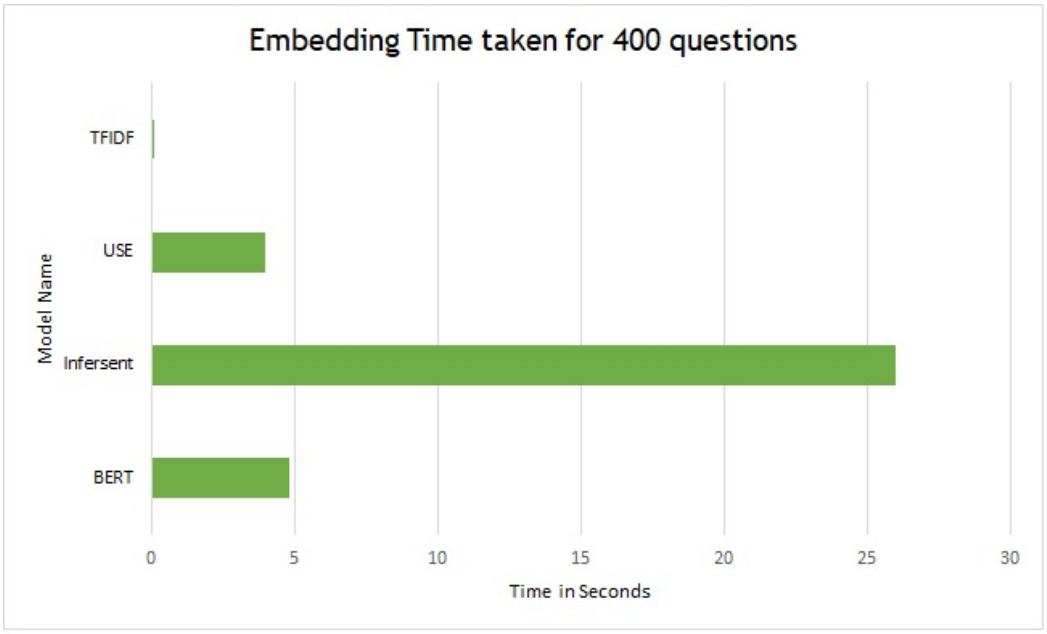

Figure 4.2: Embedding Time Comparison between Different Models

2nd and 3rd position goes to USE and BERT respectively and Infersent took the highest amount of time to embed the questions. BERT embedding took about close to 5 seconds to embed about 400+ questions. It is not desirable for a chatbot to spend 5 seconds to find an answer of a simple FAQ question from a list of only 400+ questions. The situation will be worse if we consider the FAQ data set size in thousands. That is why we came up with the idea of pre-storing the embedding vectors into a database. The details are displayed in the following section.

4.5 Answer Retrieval Time Reduction For Type II Answer

As we mentioned on the previous section, BERT embedding takes about 5 seconds for vectorizing 400+ questions. To speed up this process, we have chosen MongoDB as the database. We can not save numpy.ndarray of vectors directly into MongoDB. So, we converted the numpy.ndarray of vectors into BSON format and stored them in the database. Finally, when our system is asked a question by a user, it retrieves all the embedding vectors from the database, converts them again back to numpy arrays, and computes the similarity. This approach made a huge difference on retrieval time. As we can see from figure 4.3 that by using pre-stored embedding vectors, we were able to reduce the average answer retrieval time in EagleBot (considering searching from both type II and type III answer) from 8.5 seconds to 4.5 seconds.

4.6 Document Retrieval Accuracy Comparison

For checking how our ElasticSearch based document retriever performs, we first extracted all the URL links from Georgia Southern University sitemap.xml. Then, from those links, we extracted all the raw texts from about 28,000+ web pages in Georgia Southern University's website. We used a library called BeautifulSoup for the text extraction part. After the text extraction was done, we started indexing all those 28,000+ documents

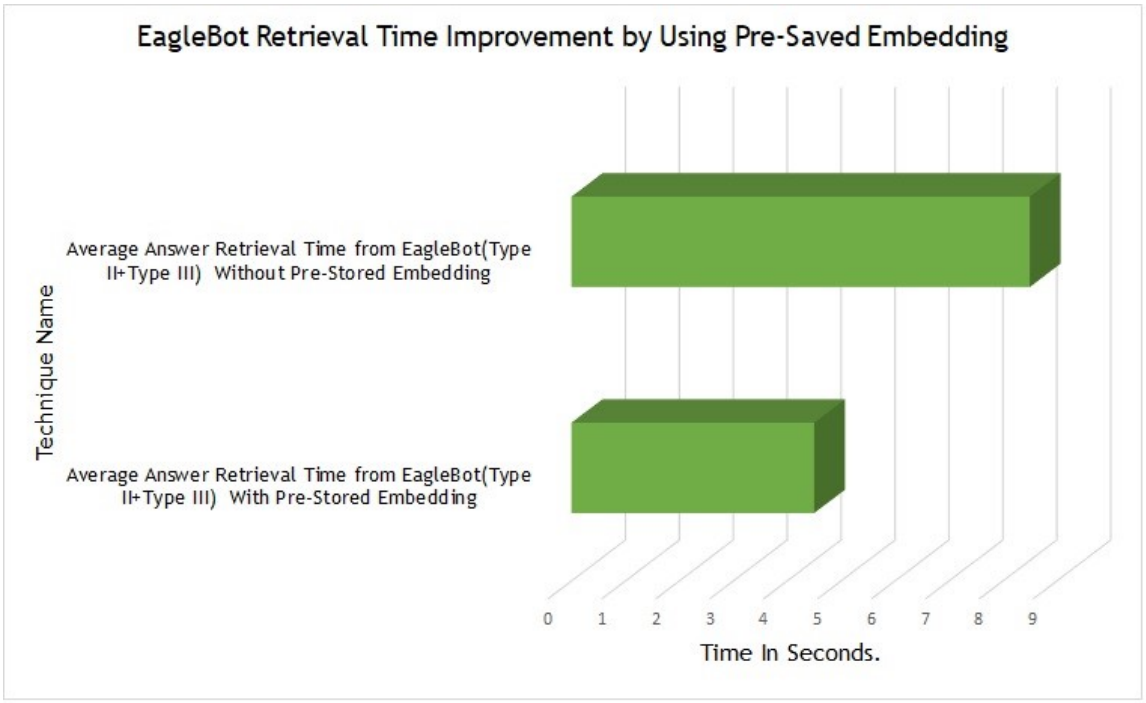

Figure 4.3: Answer Retrieval Time Improvement on EagleBot After Using Pre-Stored Embedding Vectors

Model Name	**Embedding Time (In Seconds.)**
BERT Sentence Embedding	4.8
Infersent Sentence Embedding	26
USE	4
TFIDF	0.1

Table 4.2: Embedding Time Comparison between Different Models (For 400+ Questions)

Desired Document Index	**ElasticSearch (%)**	**GoogleSearch (%)**
1st	44	64
2nd	20	24
3rd	12	8
4th..10th	0	0
Total	76	96

Table 4.3: Document Retrieval Accuracy Comparison

into ElasticSearch. Finally, we tested our document retrieval performance against GoogleSearch python API for this task.

It is very clear from figure 5.4 that GoogleSearch performs better than inverted-index based ElasticSearch. Using ElasticSearch we achieved 75% accuracy on document retrieval whereas GoogleSearch gave us 96% accuracy on our test data set. We can explain the reason in a few points: I) We lost a lot of text and data while extracting texts using BeautifulSoup since different web pages have different structures. Moreover, we only ex-

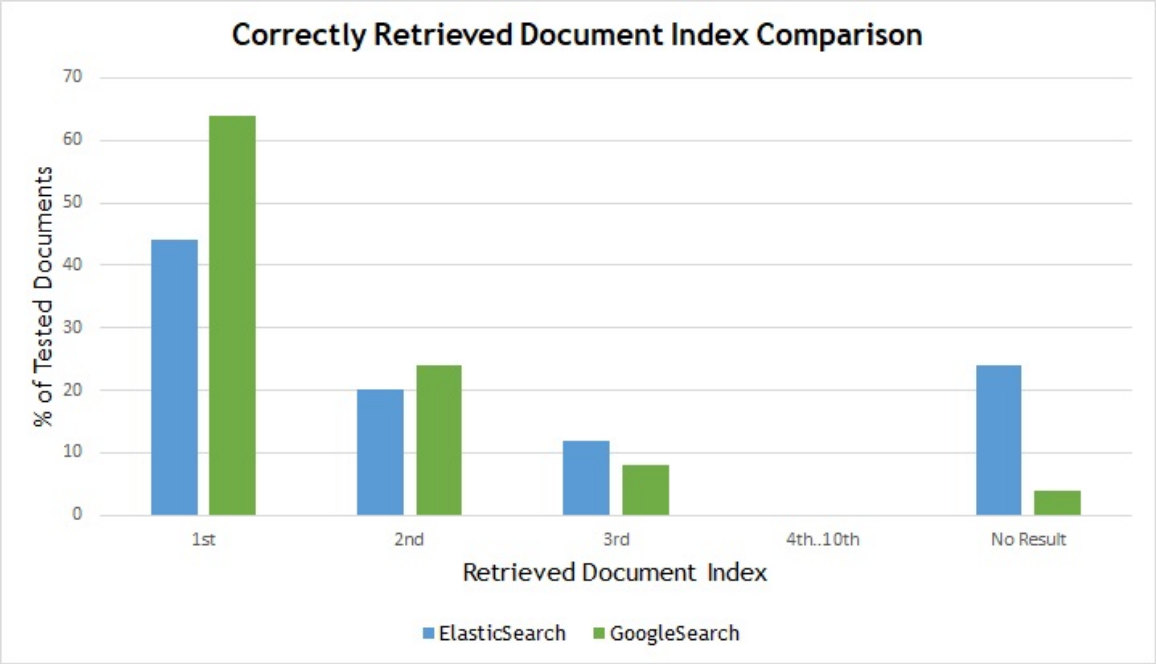

Figure 4.4: Document Retrieval Accuracy Comparison Using ElasticSearch and GoogleSearch with Desired Document Return Index

tracted texts from the main container of the pages and ignored images, sidebars, buttons, navbars, JavaScript tags, external pdf. So, we could not assure 100% pure extraction of text data. II) Google has many years of historical search data, so it is easier for Google to return a document with higher confidence than a self-indexed data storage.

4.7 Selecting *k* on Top *k* Document Retrieval

Another big challenge in this project is to decide how many documents are sufficient to ensure that we have gotten the correct answer, given a set of documents. For deciding the optimum ***k***, we saved every search result returned from ElasticSearch and GoogleSearch. We found out that, if ElasticSearch and GoogleSearch return the correct document, we can conclude with almost 100% confidence that, ***the desired document appears in the search result in between the 1st and 3rd index.*** So, in the final architecture of EagleBot, we returned the top 3 documents to find out the most probable Type III answer using our selected models. Reduction in candidate number of documents helped us in two ways. First, it helped us to get rid of extra texts and additionally it sped up the computation as it had less texts to tokenize and compute.

4.8 Document Reader (Type III Answer) Accuracy Comparison

In this experiment, we created a test dataset of 100+ questions from Georgia Southern web pages and evaluated those questions using all four techniques. The BERT-base model produced the best performance compared to the other three approaches (as shown in figure 4.5 and table 4.4) achieving about 65% accuracy. When the returned answer from the models exactly matched or partially matched with the correct answer, we counted that answer as a correct answer. Another observation from those results was, the quality of the produced answers from the BERT models were also superior to the answers produced from the BiLSTM model.

Model Name	**Accuracy (%)**
BERT-Base	65
BERT Unsupervised Paragraph Splitted	56
BERT Unsupervised Sentence Splitted	47
BiLSTM	48

Table 4.4: Document Reader Accuracy Comparison on Type III Question Using Different Techniques

Unsupervised BERT model's performance was a big surprise for us. For testing unsupervised BERT performance, we split the candidate **k** documents into multiple sentences and paragraphs. And then, we tried to find the answer from those sentences and paragraphs by using sentence embedding and cosine similarity. It turns out that, unsupervised BERT is also capable of producing good results. But, the problem is that, in unsupervised fashion, chunking a document into sentences and paragraphs is our responsibility. It is tough to decide on which basis we should partition a passage of text. It is hard to decide whether to group it using stop words or to chunk it as a passage. Sometimes users may look for a short answer and other times, they may look for a long answer. In a lot of cases, users may experience some unnecessary texts with the correct answer.

4.9 Document Reader (Type III Answer) Time Comparison

One of the biggest problems using a BERT-base model for the EagleBot is the answer retrieval time. It takes on average about 3.6 seconds for fetching an answer from 3 documents, 3.9 seconds for fetching an answer from 5 documents and about 4.4 seconds

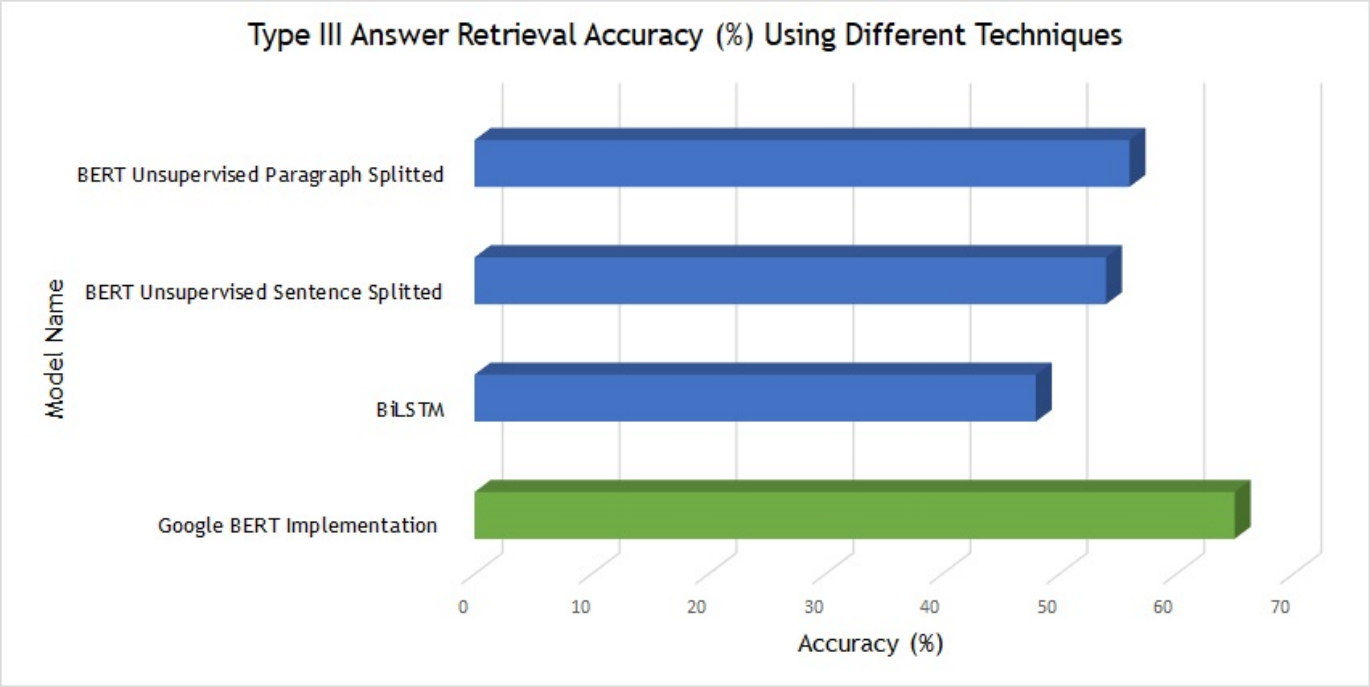

Figure 4.5: Document Reader Accuracy Comparison on Type III Question Using Different Techniques

to fetch the answer from 8 documents. Considering a chatbot system, this is quite a long time period. On the contrary, BilSTM model is blazing fast (7-10 times) compared to the BERT-base model answer retrieval time. If time is a big concern for a system, then using either the BilSTM model or unsupervised BERT will be the best option. We should mention that the retrieval time of the unsupervised BERT includes embedding time. If we use the pre-stored embedding technique that we mentioned in the section 5.4, then we can make the unsupervised BERT approaches about 10-20 times faster.

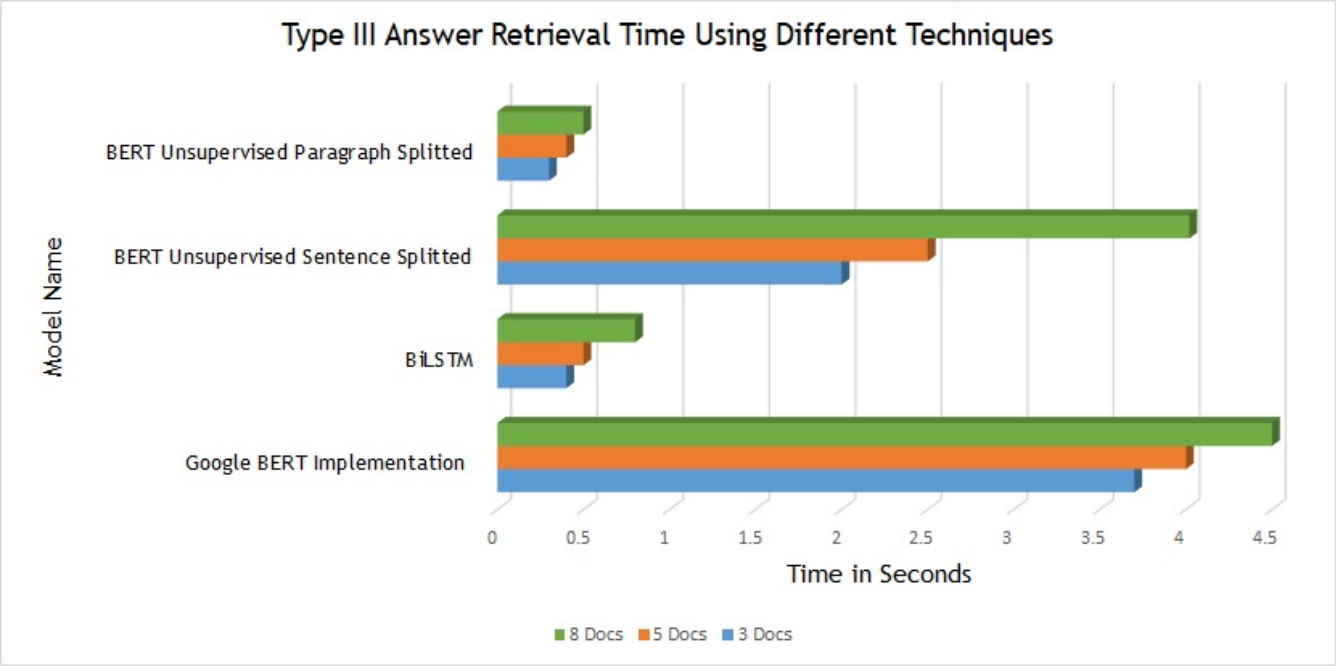

Figure 4.6: Document Reader Time Comparison on Type III Question Using Different Techniques

4.10 Cold Start Issue on Document Retrieval Models

We noticed a timing anomaly issue during the model cold-start. Whenever BERT-base/BiLSTM model gets the first request, it takes almost twice the time than a regular request. We assumed the reason behind this issue is caching. During the first request, the system started loading the model into cache memory and from the next calls the system got computational advantages from that cached memory. But, a solution to this cold-start problem is still out of our scope.

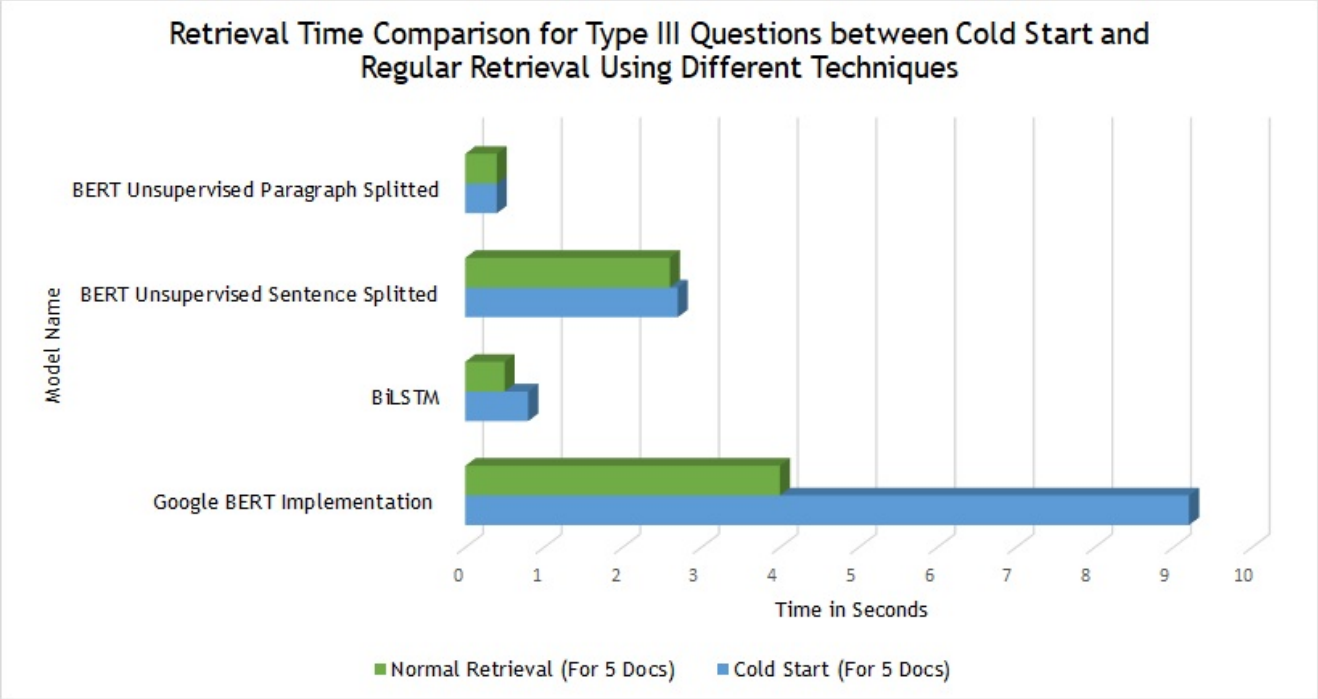

Figure 4.7: Document Reader Time Comparison on Type III Questions between Cold Start and Regular Retrieval

4.11 DIALOGFLOW DEFAULT TIMEOUT ISSUE FIXING

We have used Dialogflow as our Natural Language Understanding toolkit for detecting the user intent and extracting the important pieces of information from a question. But, Dialogflow has one major disadvantage. Currently, the default timeout in Dialogflow is 5 seconds. This means, if the answer does not come back from the server within 5 seconds, Dialogflow automatically will send a default timeout message, though the processing of answer retrieval will be still in process. For solving this 5 seconds timeout issue, we propose a few solutions:

I) While indexing the documents, the unnecessary texts from a document should be trimmed. So, when BERT model will try to tokenize the texts on those documents it will have less tokens to work with. That means, the BERT model will take less time than the larger documents.

II) Smaller $\boldsymbol{k}$ should be selected while returning top $\boldsymbol{k}$ documents from the document retriever module. This is because less number of candidate documents means less time in vectorization. During our testing, when we decreased the most probable document number $\boldsymbol{k}$ from 8 to 2-3, we achieved a better result on avoiding the 5 second default timeout issue.

III) For overcoming the drawback of default timeout issue, we have developed one independent web UI where we can control our default timeout. But, one problem with this approach was now we could not use Dialogflow's sdk to detect intent and entity. So, we needed to build the whole NLU (Natural Language Understanding) toolkit from scratch.

IV) If we still want to take the advantage of the Dialogflow NLU engine and avoid 5 seconds timeout, we can introduce threading techniques. For the test purpose, we have used a Multiprocessing package from python. Our proposed algorithm is below:

```
Start 2 threads, 1st Process tries to fetch the answer using Dialogflow & the 2nd
  process tries to fetch the answer without using Dialogflow api;
while True do
    run 2 processes simultaneously;
    if wait Time is more than 2 seconds then
        kill the 1st Process;
        continue running the 2nd Process and return result from this process;
    else
        return result from 1st process;
        break;
    end
end
```

Algorithm 1: How to utilize Dialogflow NLU sdk in parallel to EagleBot server

The algorithm basically says, we will start two threads simultaneously to utilize parallel work both with and without the Dialogflow module. And, if the answer does not return in 2 seconds from the Dialogflow supported module (If the answer is of type I or type II, it is guaranteed to return in this time frame), we will terminate this process and continue on process 2 which is independent of the time limit. By this way, we can always assure the default timeout avoidance utilizing the full potential of both modules.

CHAPTER 5

FUTURE WORK

We have got our best answer retrieval result using the BERT-base model. But, can we do better? Here are a few things that we can try to make EagleBot better.

Because of resource and memory limitation, we had to rely on the BERT-base model which has 12 layers and about 110M parameters. On their first release, Google has also released a BERT-large model which is almost 3 times heavier and larger than the BERT-base model. According to the BERT original paper, it is actually possible to achieve a much better result using the BERT-large model than the BERT-base model (Devlin et al., 2018). With ample resources, it will be interesting to observe how BERT-large will enhance EagleBot's current answer retrieval performance.

On May 31st, 2019 the researchers from Google came up with another improvement on top of the BERT-large model. On that work, they have introduced a new technique called ***Whole Word Masking*** with which they claimed to achieve better performance than the previous BERT-large model. In our future work with EagleBot, we are interested to work with this model.

One study has shown that it is possible to achieve better accuracy on question-answering tasks (they have used SQuAD 2.0) by using ensemble techniques (Zhou, 2019). Ensemble methods have given them better results than testing the model only using BERT-base. They have used several BERT models (cased/uncased, base/large). For each question, they output the n-best predictions made by each model along with the probability (after post-processing). After that for each proposed prediction, they add up the probability from each model together and finally output the prediction with the highest probability. They also incorporated BiDAF (Bi-Directional Attention Flow) with a lower weight impact into their ensemble model. In our future work with EagleBot, we tend to try a combination of BERT models with BiLSTM to test the answer retrieval performance.

Yang et al. in their recent paper have shown that they have achieved greater success when they have used paragraphs instead of feeding the whole size documents into the BERT-base model (Yang et al., 2019). They have shown that paragraph retrieval has outperformed article retrieval by a large margin as articles contain many non-relevant sentences that serve as distractors to BERT readers. But, when we go into sentence-level granularity, it also does not perform well because they often lack the context for the readers to identify the answer span. For EagleBot we have used document-level indexing. So, it can be an interesting experiment to see whether EagleBot will perform better by using paragraph-level indexing too.

Compared to the other modules, it is unfortunate to mention that our Document Retriever module (ElasticSearch based) is the weakest one in the whole pipeline. ElasticSearch returns a document based on a technique called ***Inverted Indexing***, which is in other words, a frequency based retrieval technique. So, if there are too many documents with similar or higher frequencies than the expected ones then there is a huge chance of wrong retrieval. Unfortunately, the other two notable document retrieval techniques (Yang et al., 2019) (Chen et al., 2017) are also frequency-based retrieval techniques. Another interesting future work would be developing a new retrieval technique, which considers semantic relations in text documents.

In addition, scalability is a huge concern while designing a chatbot. Due to the memory issue, we could not try BERT-large model or multiple BERT models at the same time. We were also unable to test scalability for EagleBot using BERT-base model. Therefore, in our future release version of EagleBot, we will consider these challenges.

CHAPTER 6

CONCLUSION

Though EagleBot suffers from a few issues like longer answer retrieval time for the type III questions, 5 seconds default timeout from Dialogflow and scalability, it is successful considering our initial goal. This work shows that it is possible to build a unified architecture for supporting several types of questions asked on a specific domain, except considering type I question answering (because type I is strongly domain-specific) it is possible to replicate this whole system into an entirely different domain with careful execution. Also, this system has a big promise on many other domains as most of the big organizations and companies need automation through a chatbot to handle huge traffic on their site. Our experiments have shown that BERT sentence embedding can be very successful in tasks like building a semantic search engine, FAQ search engine, text classification, and many more downstream NLP tasks. Our experiments on document retrieval modules have shown that there is still a lot of room for improvement on this part. For the document reader part, we can strongly rely on BERT models. Though we could not support bigger and stronger models from BERT due to our memory constraint, we consider this task as our future work. Even unsupervised approaches using BERT embedding on sentence and paragraph split documents have shown promising results both timing and accuracy wise on type III answer retrieval tasks. With more time and more training data, we can hope for a more accurate and faster EagleBot in the near future.

BIBLIOGRAPHY

Petter Bae Brandtzaeg and Asbjørn Følstad. Chatbots: changing user needs and motivations. *Interactions*, 25(5):38–43, 2018.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

J. Clement. Number of monthly active facebook users worldwide as of 2nd quarter 2019 (in millions), 2019.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

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

Ahmed Fadhil and Adolfo Villafiorita. An adaptive learning with gamification & conversational uis: The rise of cibopolibot. In *Adjunct publication of the 25th conference on user modeling, adaptation and personalization*, pages 408–412. ACM, 2017.

Donghui Feng, Erin Shaw, Jihie Kim, and Eduard Hovy. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 171–177. ACM, 2006.

Ashok K Goel and Lalith Polepeddi. Jill watson: A virtual teaching assistant for online education. Technical report, Georgia Institute of Technology, 2016.

Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, Alessandro Micarelli, and Joeran Beel. Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation? 2019.

Carmen Holotescu. Moocbuddy: a chatbot for personalized learning with moocs. In *RoCHI*, pages 91–94, 2016.

M Naveen Kumar, PC Linga Chandar, A Venkatesh Prasad, and K Sumangali. Android based educational chatbot for visually impaired people. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–4. IEEE, 2016.

Lindsay C Page and Hunter Gehlbach. How an artificially intelligent virtual assistant helps students navigate the road to college. *AERA Open*, 3(4), 2017.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR*, abs/1806.06259, 2018. URL `http://arxiv.org/abs/1806.06259`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

Bhavika R Ranoliya, Nidhi Raghuwanshi, and Sanjay Singh. Chatbot for university related

faqs. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1525–1530. IEEE, 2017.

Bayan Abu Shawar and Eric Atwell. Different measurements metrics to evaluate a chatbot system. In *Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies*, pages 89–96. Association for Computational Linguistics, 2007.

Aaron Smith and Monica Anderson. Social media use in 2018. `https://www.pewinternet.org/2018/03/01/social-media-use-in-2018/`, 2018.

Han Xiao. bert-as-service. `https://github.com/hanxiao/bert-as-service`, 2018.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *CoRR*, abs/1902.01718, 2019.

Wen Zhou. Ensemble bert with data augmentation and linguistic knowledge on squad 2 . 0. 2019.