Revolutionizing Healthcare Delivery through a Medical Chatbot System: Using BioBert and GPT-2

By Arupa Banerjee[1], Michael Soukup[2], Swati Sundar[3] & Wicky Woo[4]

March 11th, 2023

[1]Arupa.banerjee@gmail.com

[2]michaeltsoukup@gmail.com

[3]sundar.swati@gmail.com

[4]wicky818@gmail.com

Abstract

Modern healthcare is a slow, meandering, and inefficient process from the onset of patient inquiry to provided care. One solution to this problem is a medical chatbot. A chatbot that mimics a conversation with a healthcare provider, is capable of provider lookup, and one that can book patient appointments offers the ability to usurp the incumbent process one currently follows for obtaining medical care. In this paper, we demonstrate how a medical chatbot with state-of-the-art natural language processing models like BioBERT and GPT-2 paired with Microsoft Azure's suite of resources for deployment can serve as a minimum viable product for healthcare providers to streamline the process from patient inquiry to patient care. The key takeaways are: (1) A BioBERT GPT-2 solution shows promise for powering an artificially intelligent medical chatbot solution. (2) The system architecture is capable of building and deploying a feature-rich medical chatbot.

Keywords

Artificial Intelligence, Chatbot, Natural Language Processing, Healthcare, BioBERT, GPT-2, Transformers, Deep Learning, Neural Network, Encoder, Decoder, Sequence to Sequence, TensorFlow, Topic Modeling, Machine Learning, Term Frequency–Inverse Document Frequency, Latent Dirichlet Allocation, Non-Negative Matrix Factorization.

Introduction

Recent surveys have indicated that the majority of Americans are dissatisfied with the current healthcare system. For example, in the NRC Health's 2019 Healthcare Consumer Trends Report, most patients surveyed negatively perceived support staff, wait times, and billing (NRC Health, 2019). Similarly, a Kaufman Hall survey of 200 hospitals and healthcare executives found that 90% of respondents felt improving patient experience was a top priority. We believe a medical chatbot is one solution to aid the healthcare industry, which is currently falling short of providing adequate services for all. We hypothesize that a medical chatbot will improve the patient experience, increase access to healthcare services, and enhance overall healthcare outcomes.

Problems with the current American healthcare system stem from its rigidity and long lead times, leaving the majority of patients feeling they received insufficient care for high costs. This system works well for providing specialist care (Osborn et al., 2016). However, it is weak, considering the majority of illnesses experienced by patients are small-scale diseases like the common cold, headache, or abdominal pain. It is often possible to cure situations such as these without seeing a doctor by using simple remedies and following some medical advice. Further, Americans in today's society are seeking ways to better their well-being, particularly when it comes to mental health (Martin, 2022). Therefore, they would benefit from a convenient, empathetic, and trustworthy source to learn more about ways to improve their health and livelihood. We believe this is a gap that chatbots are ideal for serving.

There are many chatbots available in the area of healthcare that can provide answers to general healthcare questions, yet establishing natural communication so that the chatbot can function in a way similar to the communication carried out between a patient and a healthcare advisor is crucial to build a connection and engage the patient (Bhirud et al.,2019). Therefore, conventional chatbots need to incorporate Natural Language Understanding (NLU), Natural Language Generation (NLG), and Machine Learning (ML) to function like virtual care providers. Using such state-of-the-art NLP models, these techniques can improve communication in natural language and predict diseases based on symptoms.

In our work, we propose a way to build a chatbot to respond to patients' inquiries with the accuracy and empathy of a healthcare provider. The core of our product consists of a Sequence to Sequence (seq2seq) Generative Pretrained Model leveraging Biomedical texts like BioBERT and GPT-2 embeddings. These embeddings allow our chatbot to interact with users and provide the most relevant responses to their queries. Our architecture took inspiration from Doc Product, a Top 6 Finalist of the #PoweredByTF 2.0 Challenge ("Doc Product: Medical Q&a with Deep Language Models" n.d.). The idea of using a pre-trained BioBERT model is to extract embeddings for previously answered question-and-answer pairs and use these embeddings to do a semantic search on the existing question-answer pairs. There are two main components to our model architecture:-

1. We utilize the seq2seq task trained in a supervised fashion from previously answered question-answer pairs to extract the question and answers. 2. When the patient is not satisfied with the answer provided, we have to show similar responses previously answered by a doctor using a semantic search in an unsupervised fashion.

In order to make this chatbot publicly accessible, we chose to host our bot using the Microsoft Azure Cloud. We chose to utilize the Azure Cloud because its App Services enable us to host our chatbot applications. Additionally, Azure has resources such as the Azure Health Bot, which we can couple with our generative core to expand the functionality of our application to enable provider lookup, conduct rules-based health screenings, and assist in appointment booking. This system will not only respond to users' queries. Rather, it will extend beyond that to also serve as a means to connect users to the best possible health information, be it a simple question answered or booking an appointment with a healthcare professional.

Literature review

The following subsections present related literature central to understanding the paper's key themes. Michael Mauldin (1994) coined the term "ChatterBot" to describe conversational programs that could mimic human interaction and thereby pass the Turing Test, an experiment proposed by Alan Turing in the 1950s to assess the intelligence of computer programs in his famous article "Computing Machinery and Intelligence"(Turing, 1950). Since then, several chatbots have passed the Turing test, which involves a human judge who cannot distinguish reliably, based on conversational content, between the program and a real human.

Caldarini et al. have provided a foundation for exploring key considerations when implementing an Artificially Intelligent Machine such as a chatbot (Caldarini, Jaf, and McGarry 2022). Following their lead, we will also provide an overview of the current implementation approaches, different types of models, and evaluation methods and explore the cutting edge of chatbots in the field of healthcare.

Implementation Approaches:

There are mainly two types of chatbots: Rule-based chatbots and Artificial Intelligence (AI) based chatbots. In this section, we will discuss the distinguishing factors of these implementation approaches as well as the challenges and limitations of each.

The initial chatbots were rule-based as they are usually easier to design and implement. As a result, they lack in their capabilities due to their inability to answer complex questions. Rule-based chatbots answer users' queries by looking for pattern matches; hence, they are likely to produce inaccurate answers when they encounter a sentence with no known pattern. Furthermore, manually encoding exhaustive and complex pattern-matching rules can be difficult and time-consuming. Furthermore, pattern-matching rules are brittle, highly domain-specific, and do not transfer well from one problem to the other (Caldarini, Jaf, and McGarry, 2022). Joseph Weizenbaum's ELIZA was one of the first successful rule-based chatbots to simulate conversation with a human user using pattern matching and substitution techniques to generate responses to user input (Weizenbaum, 1966). According to Weizenbaum, ELIZA was successfully designated "intelligent" but was limited in its practical utility and should instead be classified as "reserved for the curious."

The principle behind artificial intelligence is mimicking human intelligence in a way that it can perform tasks, recognize patterns, or predict outcomes through learning from the acquired data from various sources (Yang, 2022). Contrary to Rule-based models, AI models utilize machine learning algorithms that train on an existing database of human conversations. Therefore, these chatbots are more flexible and no longer dependent on domain-specific knowledge, as there is no need to define and code new pattern-matching rules for each scenario. Therefore, it is possible to categorize AI-based chatbots into Information Retrieval based models and Generative models (Caldarini, Jaf, and McGarry, 2022).

Information Retrieval Models:

Information retrieval (IR) is the procedure of representing, storing, and searching a collection of big data to extract knowledge and relevant results that satisfy the user's needs as a reaction to a user's input query (Ibrihich et al. 2022,777-82). In Information Retrieval based models, the algorithm can retrieve the information based on the user's input. The algorithm is usually a Shallow Learning algorithm; however, it uses rule-based algorithms or Deep Learning in certain cases. Based on a pre-defined set of possible answers, the chatbot processes the user query and based on this input, it picks one of the answers available in its set. A database of question-answer pairs usually forms the knowledge base for this kind of model. The output returned to the user is the answer paired with the selected question among those present in the chat index, created from the question-answer dataset. IR-based approaches are great for use cases with large sets of potential questions for which rule-based implementations would become difficult. Andreas Lommatzsch and Jonas Katins explain that these approaches are implemented based on an inverted index, enabling the efficient matching of user questions with a massive set of texts (Lommatzsch and Katins, 2019). High result precision is reached by excluding stop words, reducing words to their stem, and weighting terms based on a TF-IDF scheme. In order to optimize the precision of the result set, term weights can be optimized based on relevance feedback, or query expansion techniques can be applied.

The advantage of this approach is that it uses the existing knowledge bases and FAQ lists for creating a Chabot. It would be possible to delegate the task of handling new questions to a human expert who can create a FAQ entry for the training data after answering the question. Since these models do not generate automatic responses, their results are usually satisfactory. The disadvantage of this approach is that more extended dialogs are not supported. This often results in a lack of context information, leading to reduced answer precision. One of the main limitations of this approach is that creating a knowledge base to train the model can be expensive, time-consuming, and tedious. Also, a significant amount of time and resources is required to train the system to match the user's input to the accurate answers available, which progressively becomes more difficult as the volume of the data in the training dataset grows (Nirala, Singh, and Purani, 2022).

Information Retrieval systems are less suitable for building social chatbots because they do not generate responses but retrieve them from a specific dataset. The success of social chatbots lies in their ability to respond to users' requests and establish an emotional connection with users (Shum, He, and Li, 2018). To engage the users and build a connection with them, their design must focus on both intellectual quotient (IQ) and emotional quotient (EQ), essentially by developing a personality. Users should want to engage with a social chatbot as the success metric for social chatbots is often the conversation turns per session (CPS). In addition, they must be useful and empathetic (Zhou et al., 2020).

Generative Models:

Generative models do not use responses that are already defined. Instead, as the name suggests, they generate new responses on demand (Adamopoulou and Moussiades, 2020). Generative-based models take inspiration from machine translation techniques. Similar to translations, Generative models learn user input and respond with an output similar to translating input in one language and outputting the exact phrase in another. Thus, these models can create entirely new sentences to respond to users' queries. These models still need to be trained on a vast corpus to learn sentence structure and syntax, and the outputs can somewhat lack quality or consistency. However, an evaluation revealed that over 40% of the requests are emotional. A generative model-based system is about as good as human agents in showing empathy to help users cope with emotional situations. These models outperform information retrieval systems based on human judgments and automatic evaluation metrics (Xu et al., 2017).

Types of Generative Models:

Sequence to Sequence (seq2seq) models have become the industry standard for chatbot modeling and have laid the path to Neural Conversational modeling (Caldarini, Jaf, and McGarry, 2022). The initial application of this framework involved neural machine translation and archival (Sutskever, Vinyals, and Le, 2014). In recent years, researchers have also started applying the Seq2Seq framework for building a deep NLP-based Chabot; most of this study involves an open-domain dataset that enables the Chabot to converse with humans in natural language. These models are composed of two Recurrent Neural Networks (RNN), an Encoder and a Decoder. An "encoder" RNN reads the source sentence and transforms it into a rich fixed-length vector representation, fed to the initial hidden state of a "decoder" RNN that generates the target sentence. The input sentence of the Chabot user becomes the input of the Encoder, which processes one token at a time in a specific hidden state of the RNN. The final state represents the intention of the sequence and is called the context vector. The Decoder takes the context vector as its input and generates another sequence (or sentence) one word at a time. Through backpropagation, the model learns by feeding in responses and questions. The model trains to maximize the cross entropy of the correct sequence given its context. During inference, given that the true output sequence is not observed, we feed the predicted output token as input to predict the subsequent output. This is a "greedy" inference approach. Alternatively, we can apply a beam search approach by feeding several candidates from the previous step to the next. The predicted sequence can be selected based on the probability of the sequence (Vinyals and Le, 2015).

The strength of this model lies in its simplicity and generality. We can apply it to a wide range of applications like machine translation, question/answering, and conversations without major changes in the architecture. Also, it does not involve domain-specific knowledge but is an end-to-end solution that trains using different datasets, thus on different domains. However, it can work along with other algorithms for domain-specific tasks (Vinyals and Le, 2015). On the other hand, there is a risk of information loss with longer sentences due to the limitation of this architecture, which encodes the input sentences into a fixed length vector called the context vector (Wagner, 2020).

Transformers were first presented in the paper "Attention is all you need" and has been one of the most notable innovations in the field of deep learning for natural language processing (Vaswani et al., 2017). Transformer architecture is based solely on attention mechanisms, dispensing with recurrence and convolution entirely. Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t as a function of the previously hidden state h_{t-1} and the input for position t. This inherently sequential nature prevents parallelization within training examples, which becomes critical at longer sequence lengths as memory constraints limit batching across examples. This is a fundamental constraint of sequential computation. Transformers are nowadays the model of choice for NLP challenges, replacing RNN models like long short-term memory (LSTM) by differentially weighing the relevance of each portion of the input data and providing training parallelization that permits training on larger datasets than was originally achievable. This led to the development of pre-trained systems such as Bidirectional Encoder Representations from transformers - BERT (Devlin et al., 2018) and Generative Pre-trained Transformers (Radford et al., 2019)), trained with huge language datasets, such as Wikipedia Corpus and Common Crawl, and may be fine-tuned for specific applications. Even though the evolution of transformers was to answer Machine Translation challenges, they can be adapted and modified to perform dialogue modeling tasks. There have been several versions of the Transformer since then, including the Reformer and Transformer XL, designed to address specific challenges (Tay et al., 2022).

Chatbots in Healthcare

There are many chatbots available in the area of healthcare that can provide answers to general healthcare questions. However, it is important to establish natural communication so that the chatbot can communicate in a way similar to the communication between a patient and a healthcare advisor and not a bot (Bhirud et al., 2019). It requires incorporating NLU, NLG, and ML techniques into the system for conventional chatbots to function as virtual friends. These techniques make the system more communicative in the natural language and prove fruitful for counseling and predicting diseases. Several studies have been conducted in this domain and have used different ML techniques based on the desired functionality of the product. Rakib et al. (Rakib et al. 2021) review the architecture of an integrated chatbot created for mentally ill individuals. The chatbot built using a Sequence-to-Sequence (Seq2Seq) encoder-decoder architecture responds empathetically toward patients. The encoder uses Bi-directional Long Short Term Memory (BiLSTM), and they use a Beam Search decoder that performs much better, providing empathetic responses to the user with greater precision in terms of BLEU score, as compared to a Greedy Search decoder. COVID-19 further strengthened the need to access accurate, on-demand information about the disease. People were isolated and mostly relied on online information about the disease. It was challenging to find accurate information on the web, and online communities and social media provided a limited number of relevant questions and responses to choose from, and most posted questions were not promptly answered. To address these issues, Oniani and Wang (2020) proposed to develop a chatbot enhanced by neural language models that can automatically answer questions related to COVID-19 through conversational interactions. They have utilized the GPT-2 language model and applied transfer learning to retrain it on the COVID-19 Open Research Dataset (CORD-19) corpus. In order to improve the quality of the generated responses, they applied four different approaches, namely tf-idf (Term Frequency - Inverse Document Frequency), Bidirectional Encoder Representations from Transformers (BERT), Bidirectional Encoder Representations from Transformers for Biomedical Text Mining (BioBERT), and Universal Sentence Encoder (USE) to filter and retain relevant sentences in the responses. In the performance evaluation step, they found that BERT and BioBERT outperformed both tf-idf and USE in relevance-based sentence filtering tasks as evaluated by medical experts.

Pre-training large neural language models, such as BERT, has demonstrated great results in many natural language processing (NLP) tasks. However, most pre-training focuses on general domain corpora, such as newswires and the Web. There is an assumption that even domain-specific pre-training efforts can benefit by starting from general-domain language models. However, Biomedical text is very different from newswires and web text. Hence, Lee et al. (2019) introduced BioBERT, the first domain-specific BERT-based model pre-trained on biomedical corpora for 23 days on eight NVIDIA V100 GPUs.It is a pre-trained language representation model for biomedical text mining. They established that pre-training BERT on biomedical corpora is crucial in applying it to the biomedical domain by helping it to understand complex biomedical texts. Although BioBERT architecture is almost the same as BERT, the former outperforms the latter and previous state-of-the-art models in various biomedical text mining tasks by pre-training on biomedical corpora.

Evaluation Techniques:

Evaluating Chabots is challenging since human conversation attends to different goals and functions. Based on the goal of the Chabot, the metrics used to evaluate the performance will also be different. For example, a personal assistant chatbot will be assessed based on the effectiveness of the interaction and how efficiently it executes the task. In contrast, a companion chatbot will be evaluated based on its ability to engage the users in the conversation. There are two main ways to evaluate a chatbot: human evaluation and automated evaluation.

In a human evaluation, participants interact with a chatbot and evaluate the different aspects of the interaction. However, this method is computationally expensive,time-consuming, subject to bias, and not scalable. Automated evaluation metrics are more efficient regarding the resources they need and are easier to use. For example, some of the widely used metrics for Natural language processing tasks are BLEU, METEOR, and TER, which can provide important information about the textual entailment of chatbot output (Caldarini, Jaf, and McGarry, 2022).

A few research papers have used F-score and perplexity to evaluate chatbot performance (Xu et al., 2020). The F-score measures the model's precision and recall and is often used to assess information retrieval systems such as search engines and many types of machine learning models in natural language processing. Perplexity measures the ability of the model to represent the next token probability distribution accurately. These metrics are proxies for measurements of the humanness and engagingness of a model as they attempt to mimic human responses. However, all automatic metrics have flaws (Liu et al., 2016), hence might also have to be supported by human evaluation to assess the model's performance.

Data, Data Processing, and Analysis

We researched and sourced a set of medical question-and-answer data from Mr. Lasse Regin Nielsen's GitHub repository (Nielsen, 2017). The repository contains medical question-and-answer datasets from four medical resource websites, including eHealth Forum, iCliniq, Question Doctors, and WebMD. The four JSON data files contain key-value pairs of the medical question, the answer, and the tags associated with the post.

An example in Appendix A, figure A1, shows that after merging the four JSON files into a dataframe and filtering out the rows with no tags, there are 26,417 unique medical question and answer postings. We also used a sorted bar chart to understand the different types of health-related issues that the dataset has, as shown in figure 1 below.

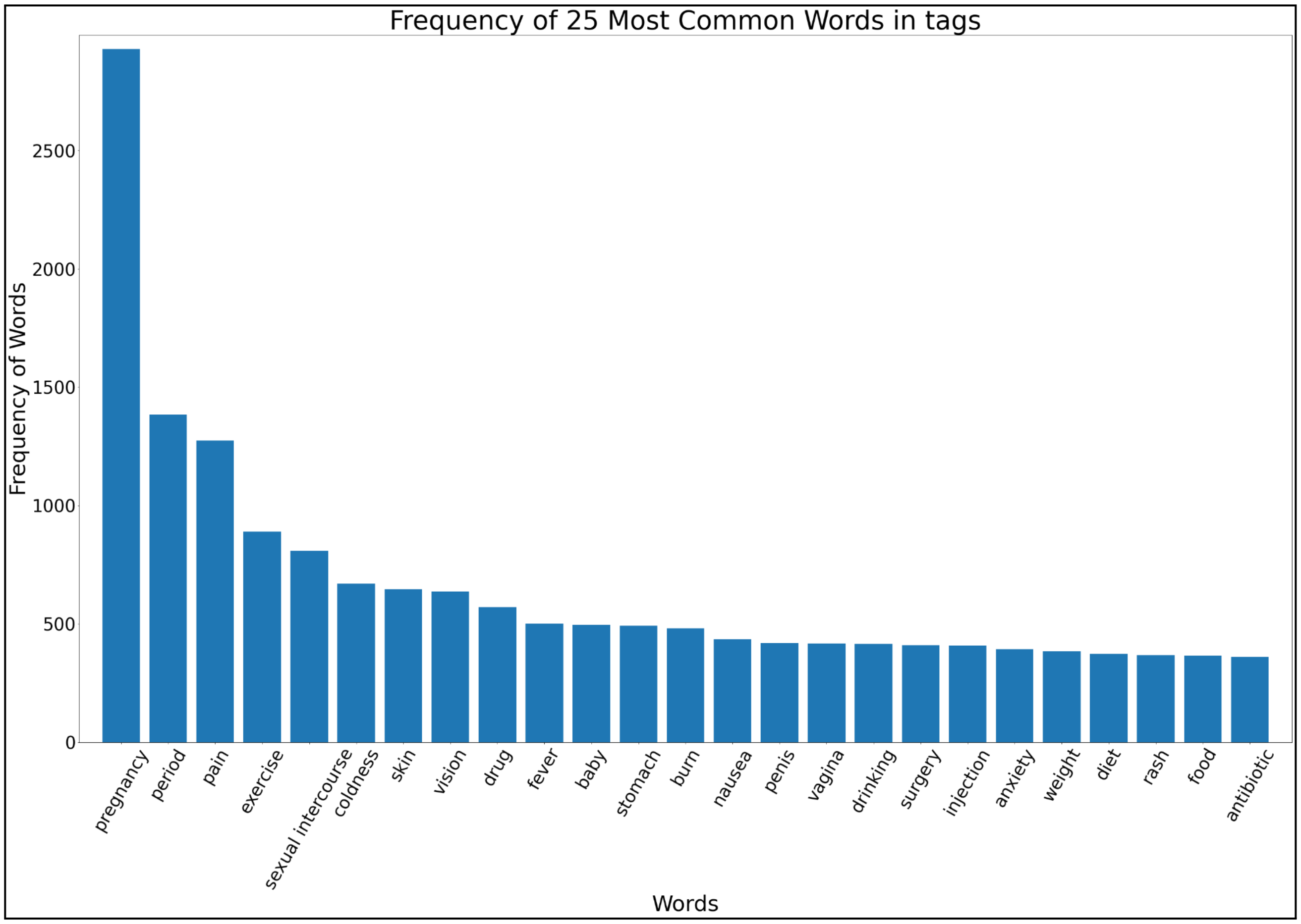


Figure 1. The most frequent tags being tagged to the questions and answers

To begin, we manipulated the data to work with the algorithms. We also cleaned the data by removing punctuation, non-alphanumeric tokens, short words, and stop words as well as converting all text to lowercase. Finally, the text was restitched back together for further processing. We kept the final dataset of questions and answers limited to 500 words as 100% of questions and 99% of all answers lie within that range.

Our dataset includes tags associated with each question and answer. These tags indicate the category to which the question and answer pair belongs. So, for every Question Q and the

Corresponding Answer A and tags T, we assigned the question-answer pair as the positive label 1. Similarly, We generate the negative label -1 by randomly picking up a tuple and checking whether the intersection between tags T and T' is null. We pair this answer A' with the original Question Q for negative samples.

After data cleaning, we performed an Exploratory Data Analysis (EDA) to search for interesting findings from the dataset. Using word count techniques and histograms, we found that most questions are fewer than 50 words, and most answers are fewer than 150 words.

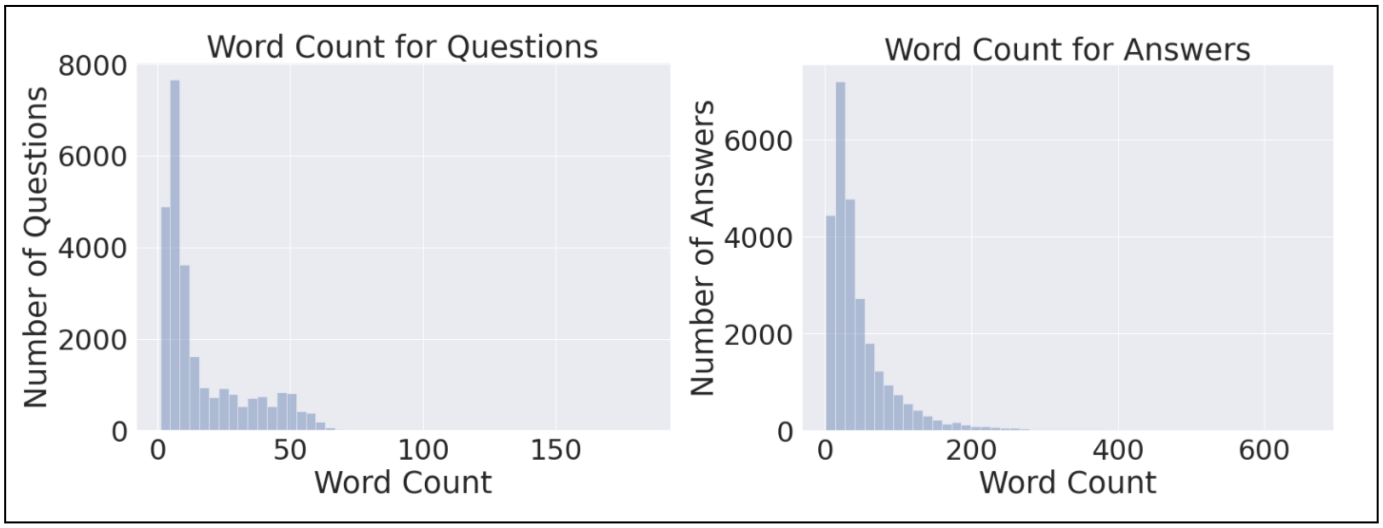


Figure 2. Word count histograms for questions and answers

Furthermore, we created word clouds for the common words in both the question and answer corpus.

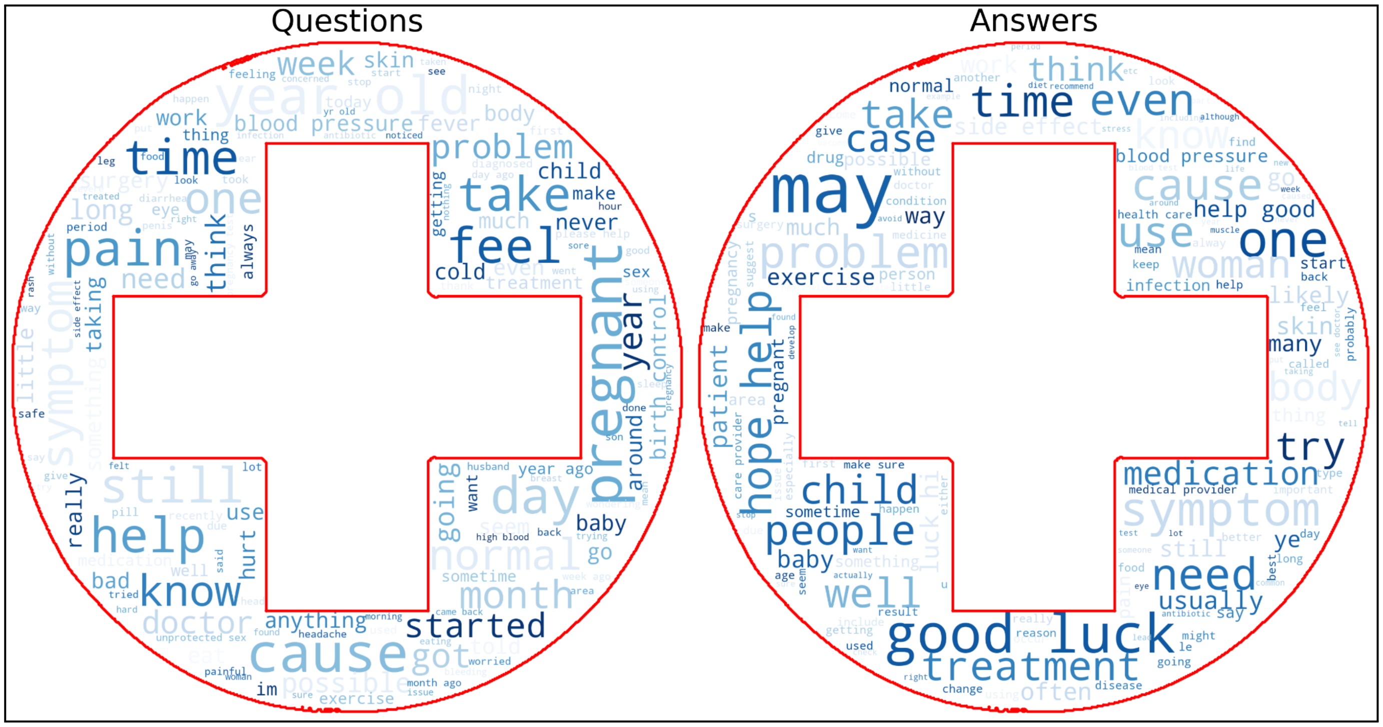


Figure 3. WordClouds for questions and answers

Methodology

Topic Modeling

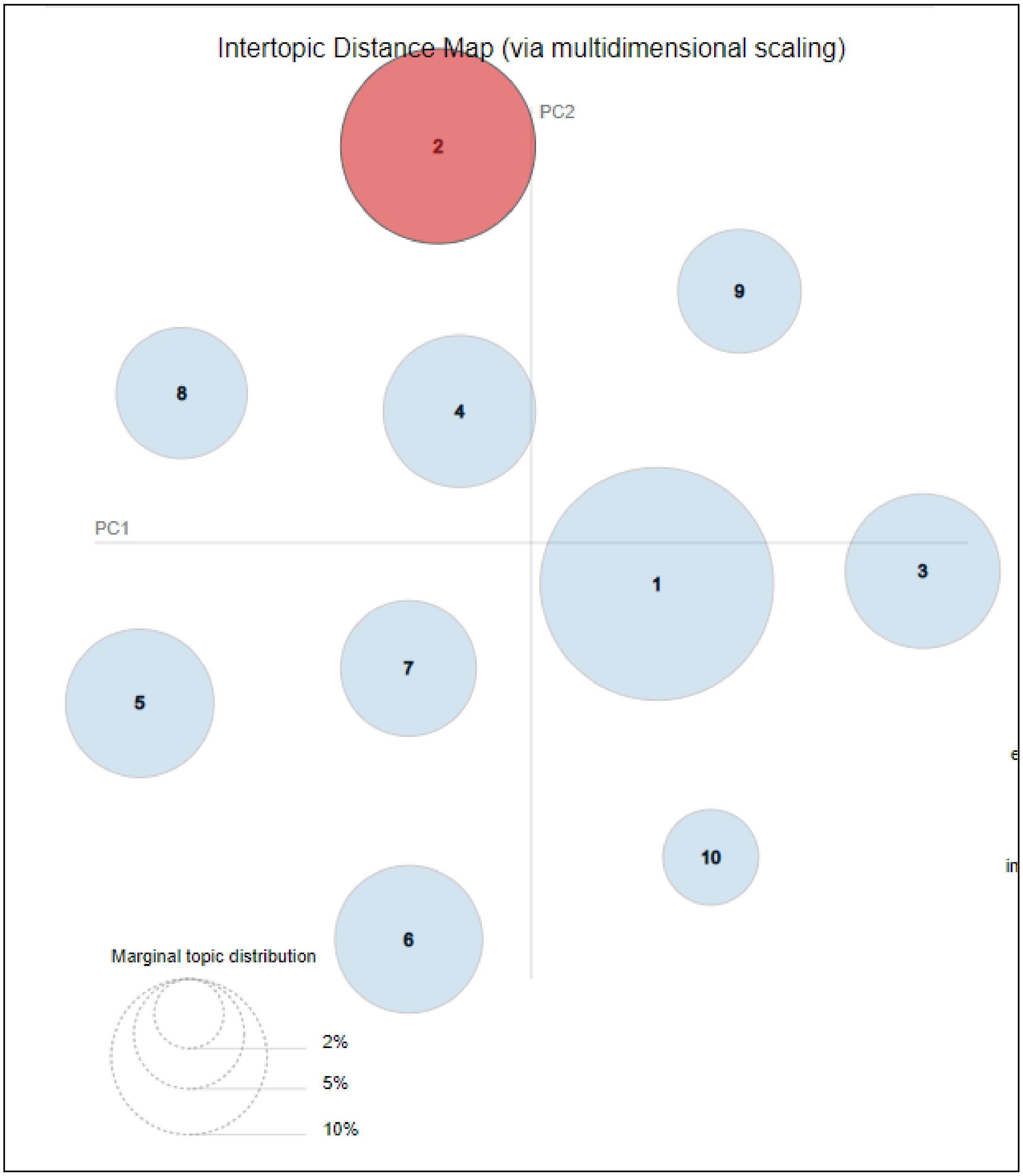


Figure 4a. Intertopic Distance Map from CountVectorizer and LDA Topic Modeling

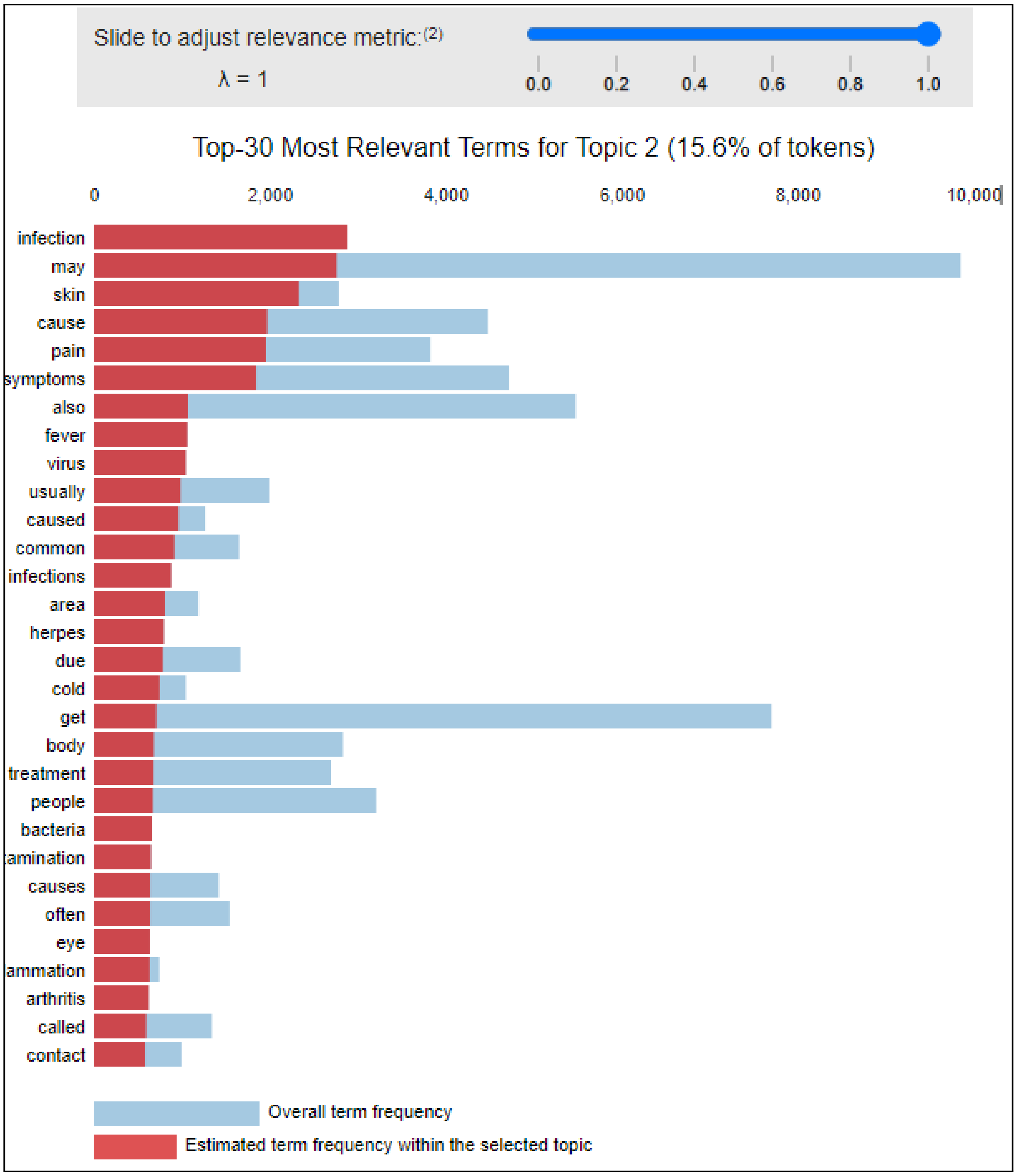


Figure 4b. The top 30 most relevant terms for topic 2 (from figure 4a)

After some data analysis, we performed a Latent Dirichlet Allocation (LDA) topic modeling for further exploration with the answers in the dataset. The cleaned text was vectorized using the CountVectorizer and fed into the LDA model with 10 topics. Please review the appendix for the results with the top 10 keywords within each topic. We also created an interactive Intertopic Distance Map with the top 30 most salient terms within each topic shown in figure 4a and 4b.

Creating the map helped us visualize where these topics are in respect to each other. Compared to CountVectorizer and LDA, we also performed tf-Idf vectorization with Non-Negative Matrix Factorization (NMF) topic modeling method to the dataset.

Experiment 1 (LSTM seq2seq):

As part of the first experiment, we trained an encoder-decoder seq2seq model using two recurrent neural networks (LSTMs), one to encode the source sequence, called the encoder, and a second one to decode the encoded source sequence into the target sequence called the decoder. We started by cleaning our corpus with regular expressions. Then, we created separate lists for input and target sequences, and we also needed to develop lists for unique tokens (input tokens and target tokens) in our dataset. For target sequences, we added ‘<START>’ at the beginning of the sequence and ‘<END>’ at the end so that our model knows where to start and end text generation. Once we had unique input tokens and target tokens for our dataset, we created an input features dictionary to store our input tokens as key-value pairs, the word being the key and the value being the index. Similarly, for target tokens, we created a target features dictionary. Features dictionary helped us encode our sentences into vectors. To decode the sentences, we made the reverse features dictionary to store the index as a key and the word as a value. To train our seq2seq model, we used three matrices of one-hot vectors, Encoder input data, Decoder input data, and Decoder output data. We have an input token from the previous time step to help the model train for the current target token. The encoder outputs a final state vector (memory) which becomes the initial state for the decoder. We use a method called teacher forcing to train the decoder, which enables it to predict the following words in a target sequence given in the previous terms. Figure 5. illustrates that the states pass through the encoder to each decoder layer (Shah, 2020). ‘Hi’, ‘how’, ‘are’, and ‘you’ are input tokens, while ‘I’, ‘am’, and ‘fine’ are target tokens. The likelihood of the token ‘am’ depends on the previous words and the encoder states. We are adding ‘<END>’ token to let our decoder know when to stop.

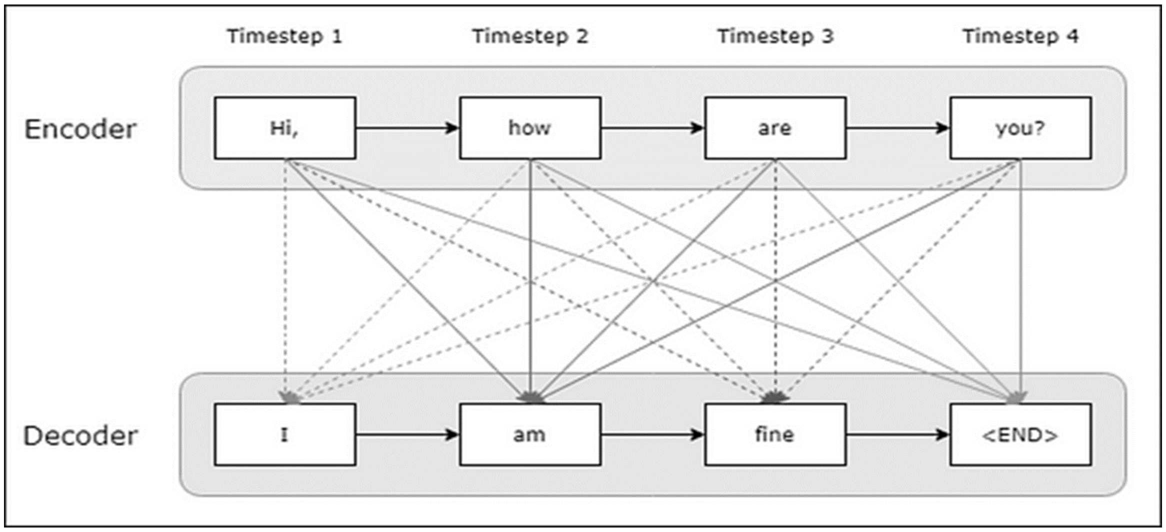


Figure 5. Encoder -Decoder model (Shah, 2020)

Experiment 2 (Transformer Encoder Decoder):

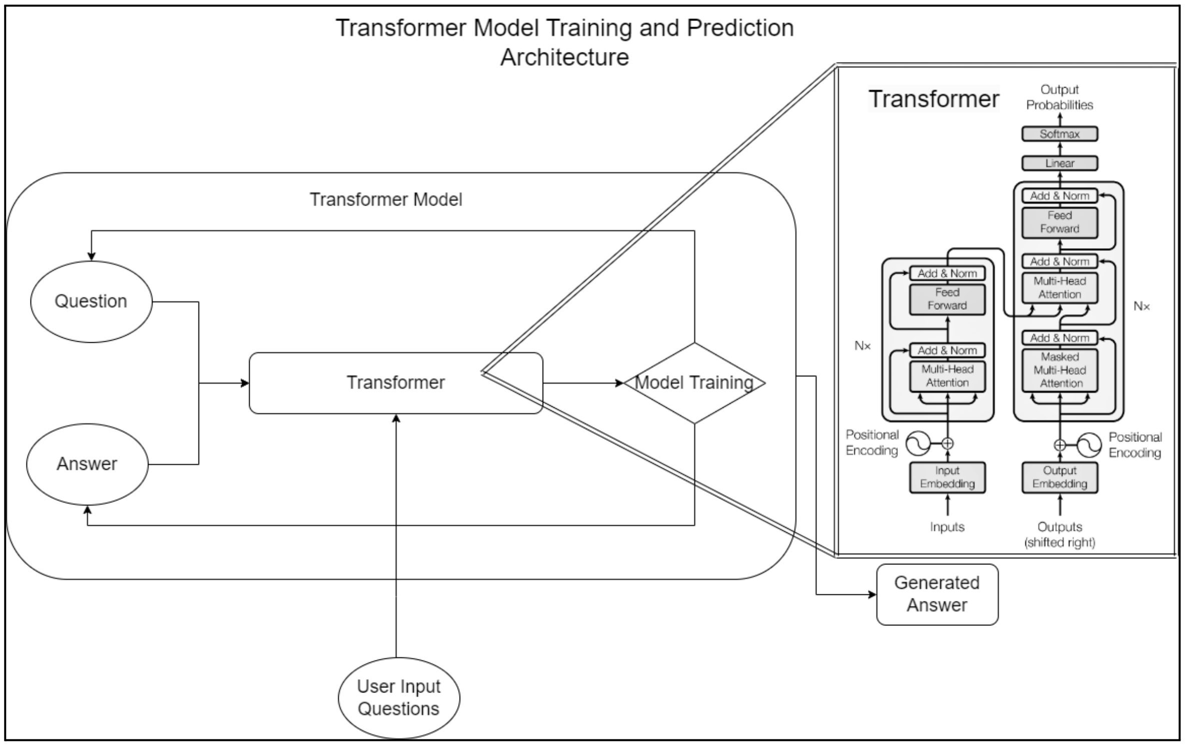
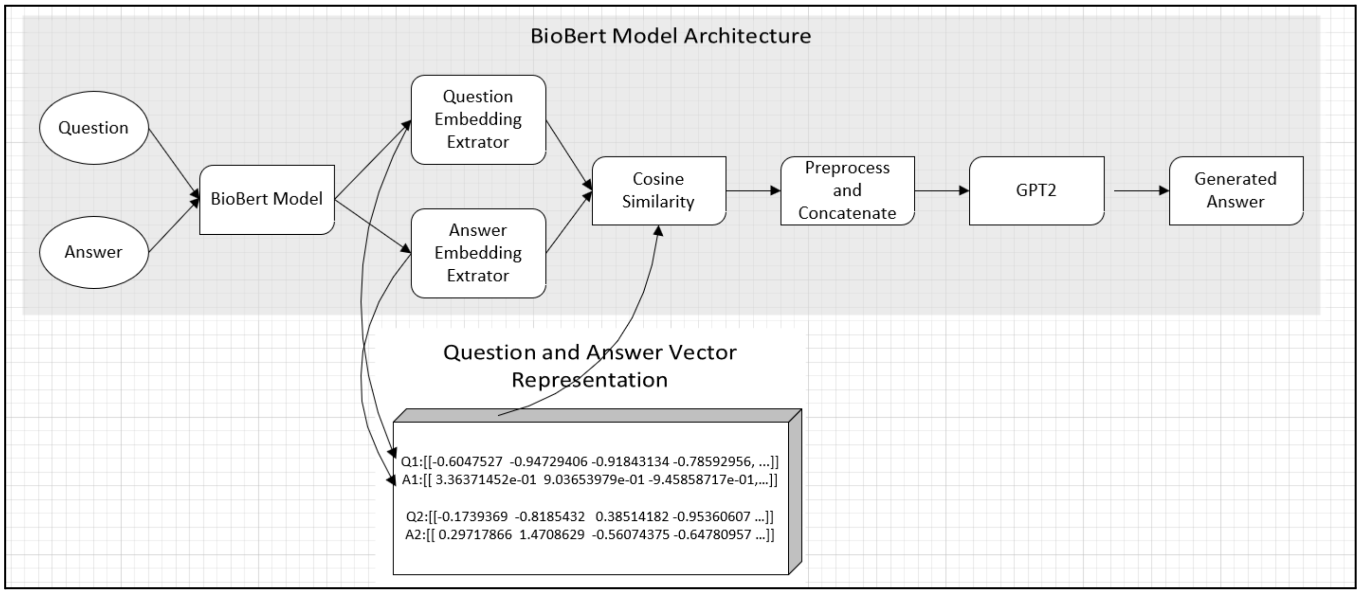


Figure 6. Transformer model architecture diagram with a traditional transformer model from the research paper “Attention is all you need” (Vaswani et al. 2017).

We performed a dataset split from the preprocessed data into a train, test, and validation set of 80/10/10. Figure 6 above shows the workflow of the transformer model training and the predictions. This is a traditional transformer implementation without any pre-trained embeddings or models. Instead, the transformer utilizes attention, positional encoding, and a decoding layer to create a model that can generate answers from user input. Bryan M. Li's blog post from TensorFlow was a big help to this implementation (Li, 2019). We were able to train the model with 100 epochs.

Experiment 3 (Pretrained BioBERT and GPT-2):


BioBert Model Architecture
Question
Answer
BioBert Model
Question Embedding Extrator
Answer Embedding Extrator
Cosine Similarity
Preprocess and Concatenate
GPT2
Generated Answer
Question and Answer Vector Representation

```
Q1:[[-0.6047527  -0.94729406 -0.91843134 -0.78592956, ...]]
A1:[[ 3.36371452e-01  9.03653979e-01 -9.45858717e-01,...]]

Q2:[[-0.1739369  -0.8185432   0.38514182 -0.95360607 ...]]
A2:[[ 0.29717866  1.4708629  -0.56074375 -0.64780957 ...]]
```

Figure 7. BioBERT and GPT-2 Model Architecture

Model development followed the same steps as transformer modeling, but we also added stratified sampling for positive and negative labels. Figure 7 shows the overall architecture of our BioBERT model. Fine-tuned BioBERT model is used to convert the text input of questions and answers to an embedding representation using the same Bert model weights. These embeddings are then input into different questions and answers layers to develop an embedding for similarity lookup. We use the same weights in the BERT layer, but the questions and answers each have their own separate layer.

Furthermore, we ensured that our quality of embeddings was maintained by not mistakenly treating similar questions and answers as negative samples. We used FAISS searches, developed by Meta for similarity search and clustering to find answers and the corresponding questions having cosine similarity to the given question in terms of embeddings and return these similar question-answer pairs in sorted order of their cosine similarity with the given question ("Indexing 1G Vectors" n.d.). We sorted and concatenated questions and answers so that the most similar question-answer pair is as close as possible to the original question. This allows us to do a semantic search task in an unsupervised learning fashion to show similar questions and answers relevant to the original question raised by the patient in case of imperfect answers generated by the Chatbot. The concatenated top similar questions and answers are then used by GPT-2 to generate an output answer.

Looking closer at the actual implementation of the BioBERT and GPT-2 Models, the BioBERT Model was loaded from the Hugging Face transformer library. In our BioBERT Model, we have two residual blocks with dropouts for each pair of questions and answers. We were able to reduce the overfitting of the model using dropout layers. We trained the model with a learning rate of 0.0005 for 5 epochs. Next, we saved the model in order to extract the question and answer embeddings. Using the saved fine-tuned BioBERT model, we extract the embeddings by passing them through both the residual blocks of the question and answer. We then store both of these embeddings along with the question and answer in order to perform a semantic search. Next, FAISS was used to find similar answers to each question in the train dataset. We performed a FAISS search based on the pre-computed Bert embeddings of each question and answer.

In addition to grouping them based on similarity, FAISS scores and sorts how similar they are. Therefore, our model can make better predictions by having more recent context. Following that, we feed the concatenated and sorted question-and-answer pairs to our pre-trained GPT-2 model. Since GPT models have a context size restriction, we used a maximum sequence length of 1024 to train our pre-trained GPT-2 model. We took the 1024 tokens post-tokenization. As the GPT-2 model predicts the next word using the context of the entire sentence, we must ensure that the model knows that when we input a sorted and concatenated Question and Answer pair for, e.g., Q2A2Q1A1Q, the model should be able to predict A based on Q2A2Q1A1Q. To accomplish this, we compute the loss mask, and while calculating the loss for GPT-2 predictions, we multiply this loss mask with the original loss so that it learns to predict the answer to the question asked by considering both the question and previous similar question-answer pairs. Moreover, we used different methods to generate GPT-2 text, such as Greedy Search, Beam Search, and top-p and top-k sampling schemes, which we will discuss in greater detail later in this paper.

Our primary goal is to create a minimal viable product of a medical chatbot using the artificial intelligence and NLP techniques described above that is publicly accessible such that users can interact with and receive medical information. We have just discussed our approach to developing the NLP engine that will power our chatbot, so now we will turn our attention to how we will package this engine into a deployed application that users can interact with.

Figure 8 illustrates the basic deployment architecture we utilize for our minimum viable product. We will spend the first part of this section unpacking the construction of this system architecture.

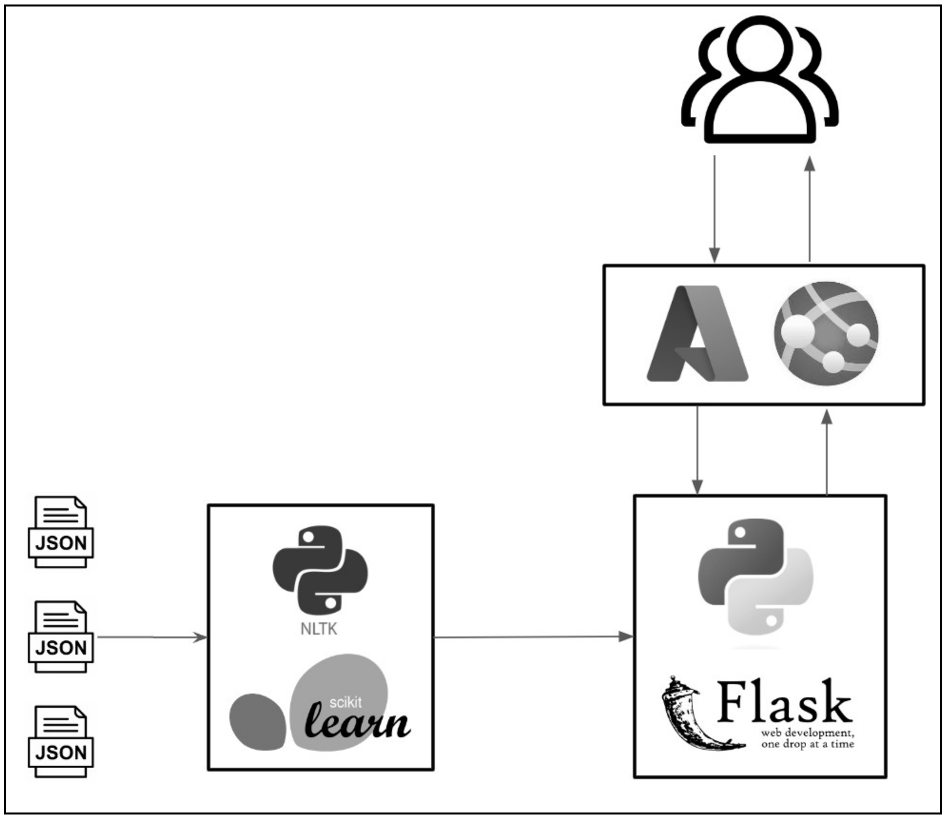


Figure 8. Deployment architecture for medical chatbot MVP utilizing Flask and Azure.

We leveraged the Flask microframework to host our application code (Flask Documentation (2.2.x) n.d.). Utilizing Flask, we are able to develop our application using a combination of Python, HTML, CSS, and JavaScript files to create a simple yet powerful application and user experience for our medical chatbot. We made this choice because of Flask's elegance and simplicity. Additionally, we believe a web application will enable us to reach the broadest audience possible, allowing our chatbot to interact with users in a digitally native way. Leveraging Flask's innate routing functionality, we created a framework for users to ask questions and receive answers from our model.

We can implement this architecture because our NLP models are portable. To illustrate the simplicity, we will now share our high-level development cycle. First, we utilize Google Colab notebooks to develop and experiment with machine learning models. Next, we save these models as pickle and h5 files for ease of portability. Finally, we load these models into our working directory and access the models with Python scripts. Our chatbot is equipped with models and user input to interact with incoming user text and answer queries on demand.

To deploy our application, we utilized Microsoft Azure Cloud as our deployment and hosting provider. We chose Azure because of the simple, cheap, and robust ability to deploy projects via their App Service. Utilizing the Basic B1 App Service plan, we can deploy our app on a server with 1.75GB memory, 10GB storage, and an SLA of 99.95% for only $12.41 monthly. Not only does Microsoft Azure provide the computational resources we require to host our application for a low cost, but the platform allows seamless CI/CD and git integration to support frequent changes and enhancements of our application during development.

Our MVP will serve as the core engine of our application. It will allow users to ask medical queries and receive thoughtful, accurate, and compassionate responses in return. While useful, a question-and-answer engine has limited practical utilization. As such, we wish to incorporate a collection of features into our application to greatly improve its utility to both individuals seeking medical guidance and medical practitioners.

The first feature we would like to incorporate in addition to our engine is the ability of our chatbot to also serve as a virtual assistant for any medical practice by connecting users to medical professionals within the organization. We can achieve this by analyzing the context of the user's discussion with our chatbot and directing the user to a medical professional in a field matching their interests. In doing so, our chatbot will perform provider lookups and help schedule appointments based on provider availability. This behavior will help convert user interaction with the chatbot to patient visits to help connect users with the best possible information and care they need. In return, we believe medical practices will see an increase in patient visits, revenue, and customer service.

Another feature we would like to incorporate into our application is the ability of our model to read medical images to aid in a diagnosis. This feature will streamline the diagnosis process for patients who require medical imaging. Instead of seeing a primary care physician, getting a referral for a medical image, completing the imaging process, and returning to the primary care physician to receive a diagnosis, our application will short-circuit this process. Utilizing our application, users can go to a medical imaging center, upload their images to our bot, and receive a diagnosis. This utility, in conjunction with the partnership with a medical practice, can enable users to take healthcare into their own hands and get answers to pressing medical questions quickly.

These additional features can significantly improve our user experience as a one-stop shop for medical information and even diagnosis. Figure 9 demonstrates how we will modify our MVP architecture to incorporate this extended functionality.

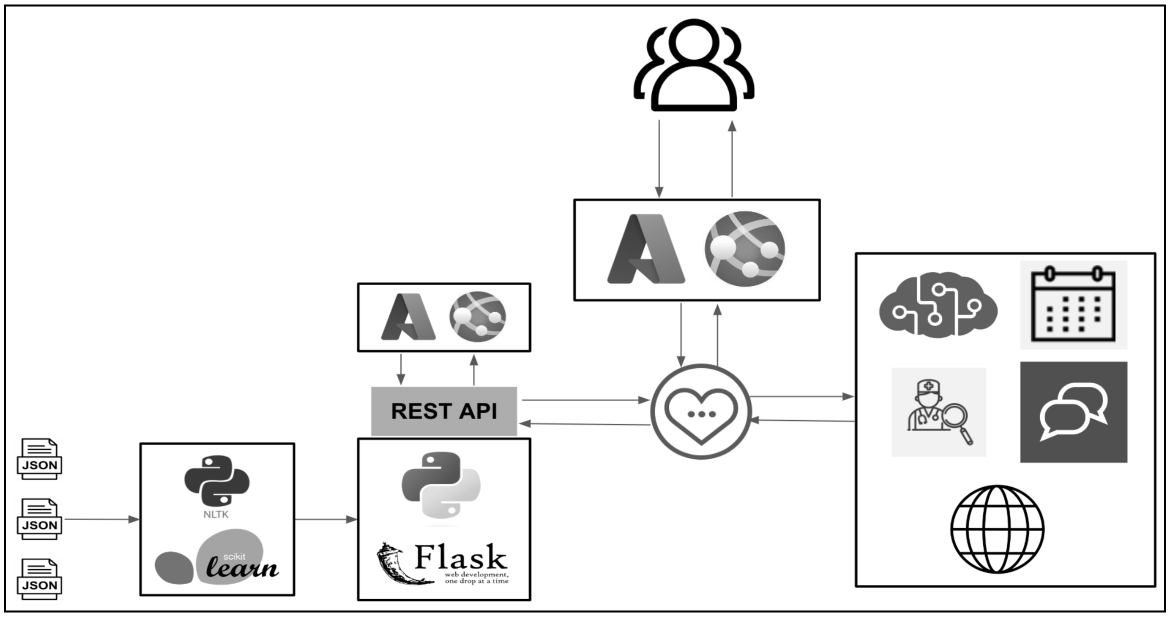


Figure 9. Deployment architecture for a medical chatbot application leveraging developed NLP models to handle user inquiry, image recognition, and Microsoft Azure's suite of resources.

Our MVP architecture is elegant and built to be flexible. When incorporating additional features, we will add to the existing architecture piecewise. To do this, we will leverage the Azure Health Bot resource, which enables conversational healthcare experiences at scale (Microsoft Azure, n.d.). This chatbot framework enables us to leverage Azure's Cognitive Services to enrich chatbot conversations with utilities such as LUIS and QnA Maker (Microsoft Azure, n.d.). LUIS is a service that determines the intent of a user's input, while QnA Maker determines the answers to a user's text. In addition, we can quickly increase the utility of our application by incorporating many conversational templates native to the Azure Health Bot, such as Covid-19 FAQs, Mental Health Screeners, and Flu Vaccination FAQs. Lastly, the Azure Health Bot can quickly enable provider lookup and appointment bookings, among other features, which help accomplish our goal of improving the patient experience, increasing access to healthcare services, and enhancing overall healthcare outcomes.

We will make minimal architecture changes to our MVP to implement these features. First, we will modify our web application to support REST API protocols to send and receive data between the Azure Health Bot. The Azure Health Bot will become our UI and conversation router. Finally, we must instantiate an Azure Health Bot instance and deploy it via Azure App Services to achieve these goals. Once these steps are complete, we will add features as time allows. For instance, we can develop and port image recognition models into our existing web application. Alternatively, we can incorporate existing Azure services, such as appointment scheduling or provider lookup, to enhance our application and the user experience.

Ethics and Responsibility

While our application can revolutionize the healthcare industry, this does not mean there are no barriers to widespread implementation. Among those barriers, the largest of which concerns the ethics of a medical chatbot and the coinciding responsibility placed on the product owners.
One of the biggest concerns with implementing a product such as this is that patients may rely on chatbots for medical advice and diagnosis in lieu of seeking medical professionals. We designed our application intending to assist medical professionals while providing easily accessible medical information to all. The intent is not to replace medical professionals but rather to be an extension of current healthcare systems. Problems arise when patients trust the medical chatbot above other medical professionals. Users of our product must understand that artificial intelligence can be accurate but is often incorrect. Patients who trust an incorrect diagnosis without seeking proper medical care could harm themselves and others with that misinformation. These risks increase greatly, considering the severity of the diagnosis. For instance, incorrectly diagnosing the common cold will have lower adverse effects compared to incorrectly diagnosing cancer. Thus, we must design and implement our medical chatbot in a way that acknowledges its limitations and clearly communicates the intent and proper use to the patients who use it.

Another ethical consideration is the impact of incorrect medical diagnoses from our chatbot. We must face this in tandem with common AI dilemmas such as bias and lack of transparency. AI outputs are only as good as the data used to train them. As such, biased training data results in biased

AI solutions. This can lead to incorrect outputs, discriminatory results, and other harmful outcomes which are difficult to predict. Compounding this issue is the lack of transparency regarding how AI models generate results. The inner workings of AI algorithms can be exceedingly complex and difficult, if not impossible, to understand. Thus the product owner must be fully knowledgeable about why certain outcomes are obtained, especially if the results provide a disruptive diagnosis based on sex, race, religion, or gender, as the product owner is accountable for the chatbot's outputs. This increases the liability placed on the product owners and is a barrier to widespread implementation.

Another consideration worth mentioning here is that our chatbot must also keep patient data private. Our bot will interact with sensitive and personal health information. Thus our bot must protect this data, keep it confidential, and store it securely. Federal and international legislation mandates these requirements. Examples of legislation our chatbot will adhere to are The Health Insurance Portability and Accountability Act (HIPAA) (Health Insurance Portability and Accountability Act, 1996); The Substance Abuse and Mental Health Services Administration (SAMHSA) (Substance Abuse and Mental Health Services Administration [SAMHSA], n.d.); The California Consumer Privacy Act (CCPA) (California Consumer Privacy Act, 2018) and the General Data Protection Regulation (GDPR) (General Data Protection Regulation, 2016). We must take these data governance considerations seriously before full-scale product deployment; otherwise, there will be serious legal, financial, and social fallout.

While creating a medical chatbot is technically feasible, as outlined in this paper, we still need to overcome significant challenges before widespread implementation. The most significant challenges are the ethical and data governance challenges accompanying an artificially intelligent engine that collects users' medical data in return for providing medical information, advice, and diagnoses. Therefore, the bot's users must understand our product's benefits and limitations. Likewise, we must repeatedly reinforce that this chatbot is an extension of the healthcare industry and by no means a substitute for professional medical care.

Results

We ran three experiments, starting with a baseline model and successively leveraging pre-trained models to evaluate and compare the results. First, we used baseline metrics like accuracy and loss to assess the model's performance. Furthermore, we evaluated the model results with a more advanced automated metric, the Bilingual Evaluation Understudy (Bleu score). This section will review the first-pass results and the techniques employed to improve performance with large language models (LLM).

Experiment 1 (LSTM seq2seq):

In our first experiment on generative-based chatbots, we trained an encoder-decoder model that uses Long Short Term Memory- LSTM for text generation from the training corpus. Figure 10 shows the model architecture of the seq2seq model.

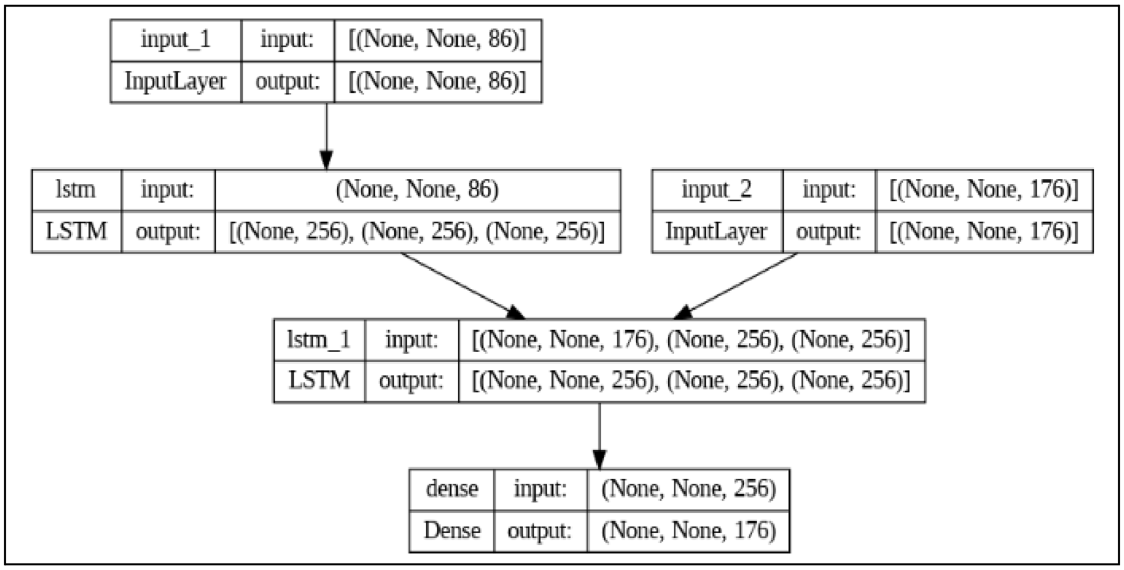

input_1 | input: | [(None, None, 86)]
InputLayer | output: | [(None, None, 86)]
lstm | input: | (None, None, 86)
LSTM | output: | [(None, 256), (None, 256), (None, 256)]
input_2 | input: | [(None, None, 176)]
InputLayer | output: | [(None, None, 176)]
lstm_1 | input: | [(None, None, 176), (None, 256), (None, 256)]
LSTM | output: | [(None, None, 256), (None, 256), (None, 256)]
dense | input: | (None, None, 256)
Dense | output: | (None, None, 176)

Figure 10. Model architecture of the seq2seq model

We have used rmsprop as an optimizer and categorical_crossentropy as our loss function. Then, we call the .fit() method by giving the encoder and decoder input data (X/input) and decoder target data (Y/label). After the training finished, we got an accuracy of 23%.

The model was trained over 1000 epochs with 839,856 trainable parameters and had a training accuracy of 23%, which is not that great, and a validation accuracy of 5%, which was even lower.

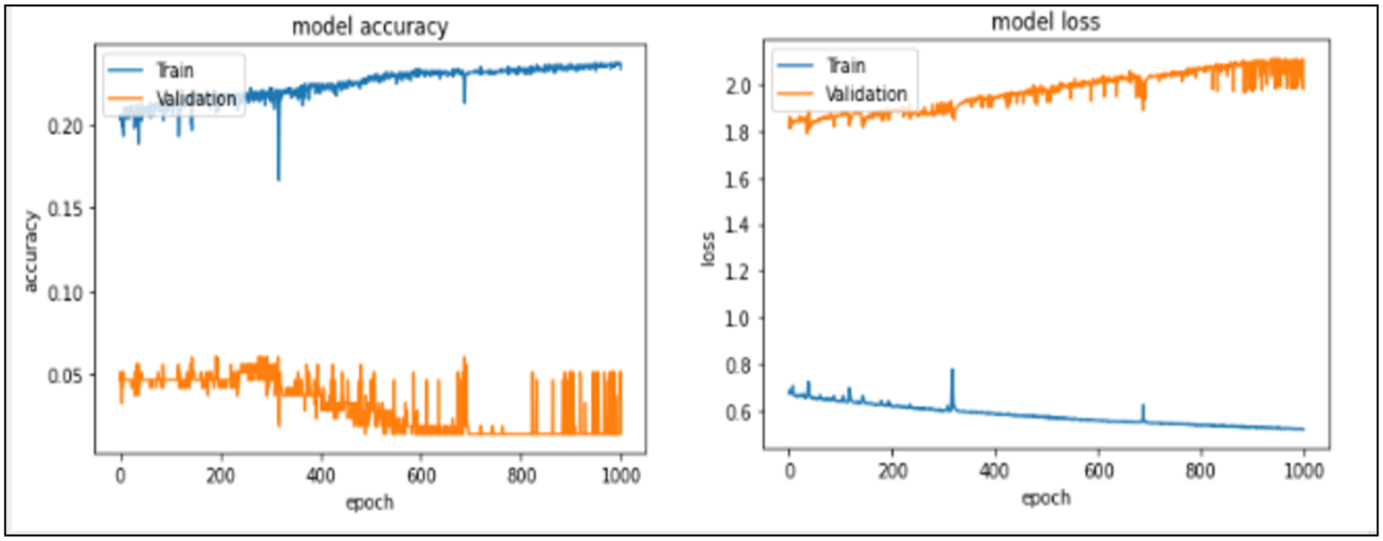
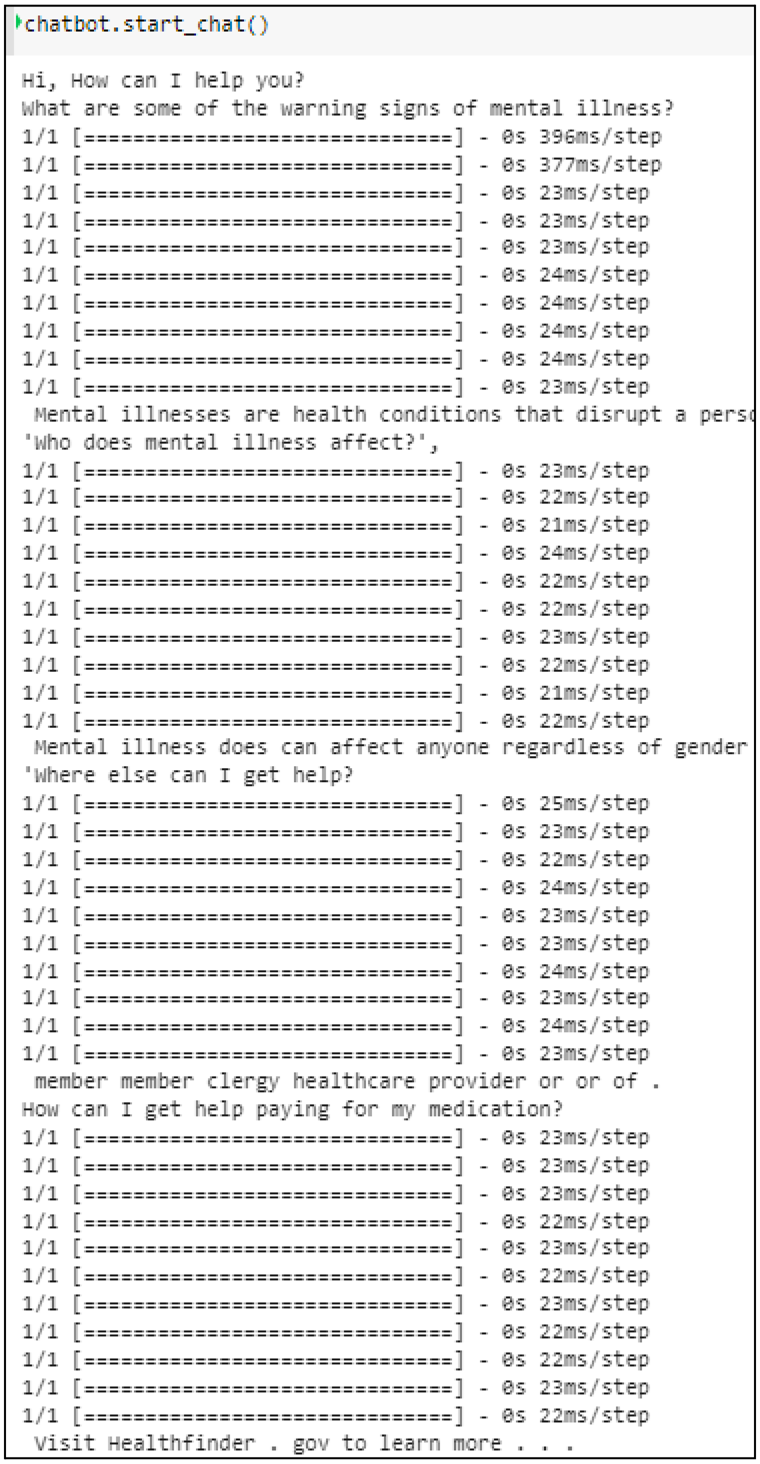


Figure 11. Training and validation accuracy of the seq2seq model

Here are some examples of the predictions it was able to generate.



```
chatbot.start_chat()

Hi, How can I help you?
What are some of the warning signs of mental illness?
1/1 [==============================] - 0s 396ms/step
1/1 [==============================] - 0s 377ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
 Mental illnesses are health conditions that disrupt a pers
'Who does mental illness affect?',
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
 Mental illness does can affect anyone regardless of gender
'Where else can I get help?
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
 member member clergy healthcare provider or or of .
How can I get help paying for my medication?
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
 Visit Healthfinder . gov to learn more . . .
```

Figure 12. Predictions made with the seq2seq model

Experiment 2 (Transformer):

For the Transformer model, the results are not too accurate. The model achieved a training accuracy of 9.6% and a validation accuracy of 8.1% over 100 epochs. You may view the learning curve in figure 13. Figure 14 shows some examples when asking questions using the predict function for predicting a response from the Transformer model after the training.

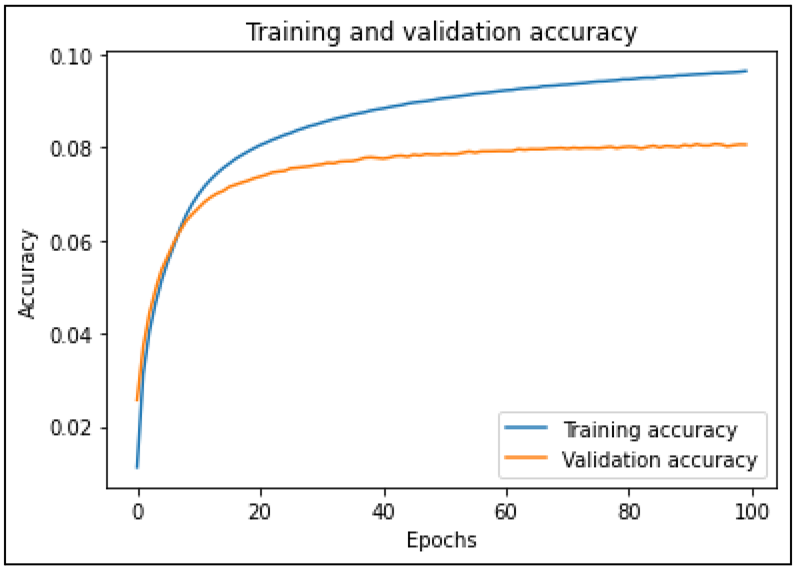
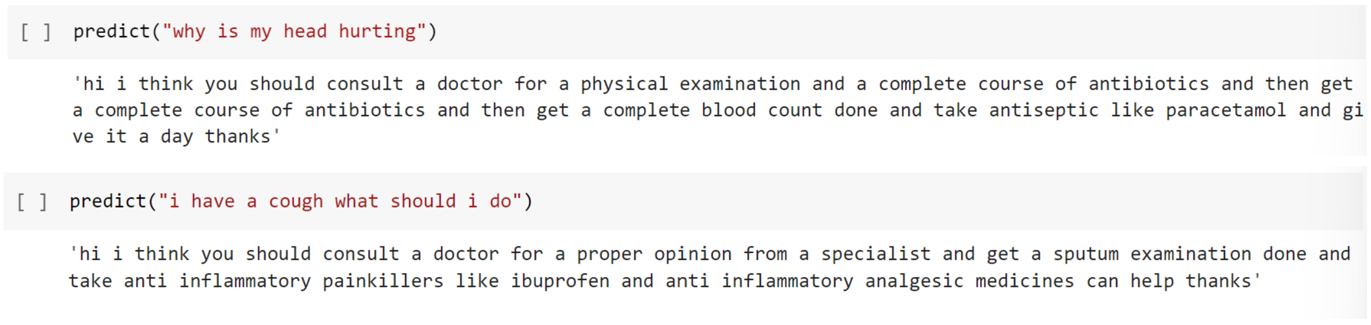


Figure 13. Training and validation accuracy using the traditional Transform method



```
[ ] predict("why is my head hurting")

    'hi i think you should consult a doctor for a physical examination and a complete course of antibiotics and then get
    a complete course of antibiotics and then get a complete blood count done and take antiseptic like paracetamol and gi
    ve it a day thanks'

[ ] predict("i have a cough what should i do")

    'hi i think you should consult a doctor for a proper opinion from a specialist and get a sputum examination done and
    take anti inflammatory painkillers like ibuprofen and anti inflammatory analgesic medicines can help thanks'
```

Figure 14. Predictions made with the Transformer model

Experiment 3 (BioBERT and GPT-2).

BioBERT achieved a training accuracy of 89% and a validation accuracy of 79% over 5 epochs. The learning curve in figure 15 shows that the pre-trained BioBERT model outperformed the conventional transformer model. Following the extraction of BioBERT question and answer embeddings, sorting, and concatenation of similar question and answer pairs, we trained the GPT-2 model over 5 epochs. Next, We evaluated the GPT-2 models generated answers using the Bleu score. Figure 16 shows the bleu score as 0.07. Furthermore, we found that GPT-2 model text generation differed significantly from Greedy search, beam search, and Top-K sampling methods in terms of their results. Figures 17, 18, and 19 illustrate the GPT-2 generated responses using Greedy Search, Beam Search with Ngram-3, and Top K sampling with Top k = 20, respectively.

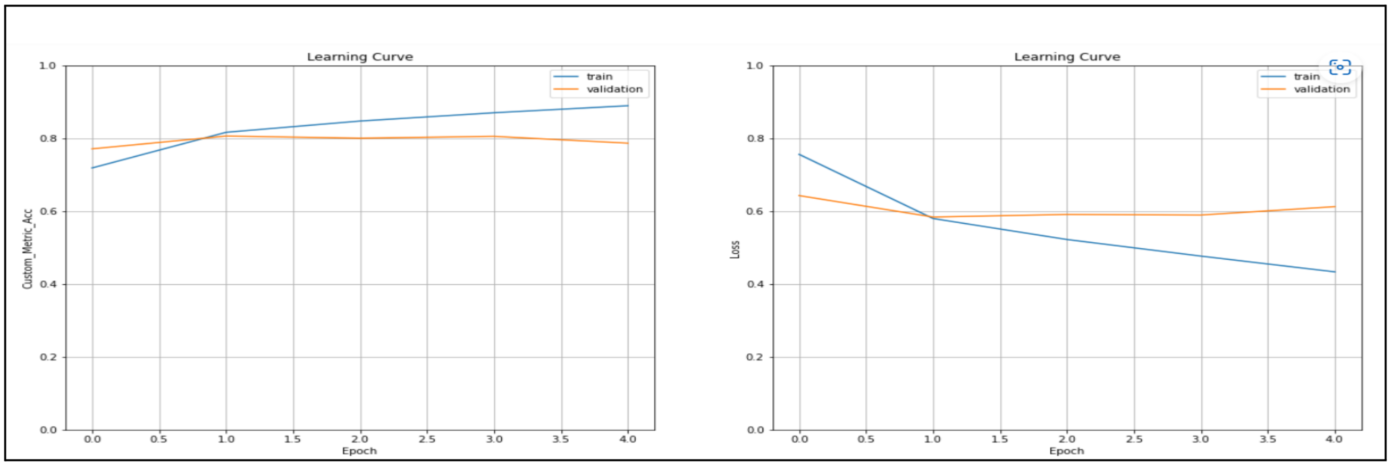
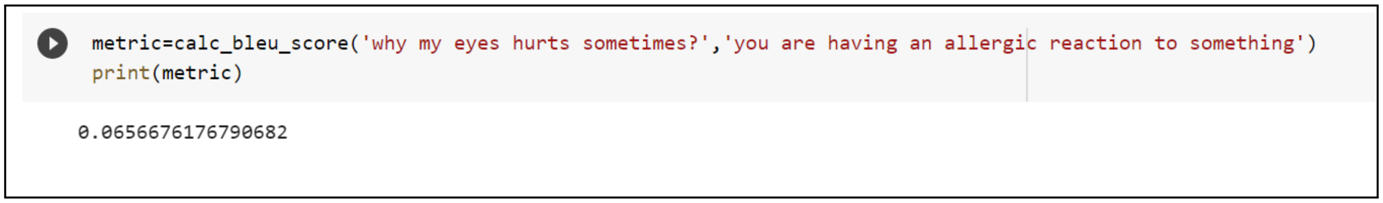


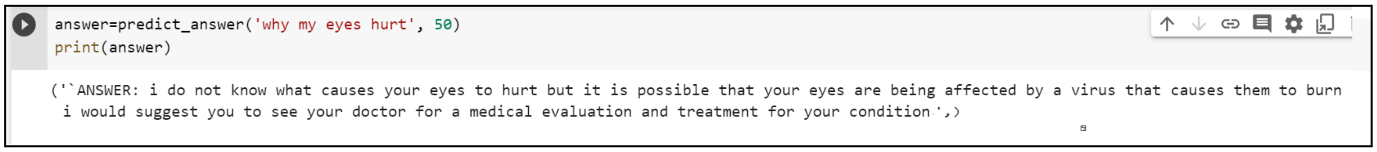
Figure 15. BioBERT Model Learning Curve



```
metric=calc_bleu_score('why my eyes hurts sometimes?','you are having an allergic reaction to something')
print(metric)

0.0656676176790682
```

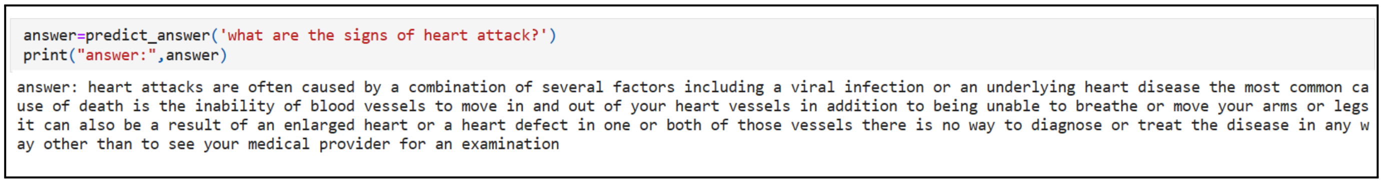
Figure 16. GPT-2 Model Bleu Score



```
answer=predict_answer('why my eyes hurt', 50)
print(answer)

('`ANSWER: i do not know what causes your eyes to hurt but it is possible that your eyes are being affected by a virus that causes them to burn
 i would suggest you to see your doctor for a medical evaluation and treatment for your condition`',)
```

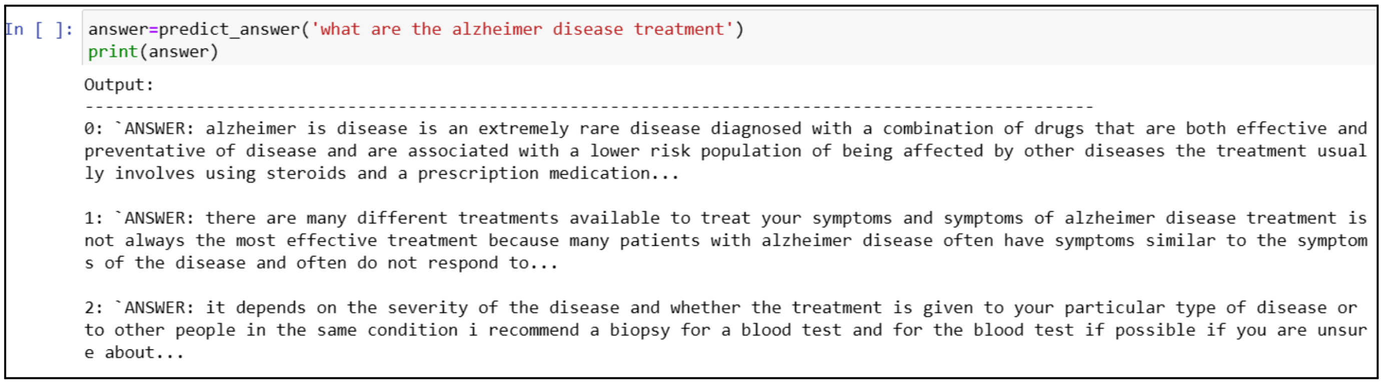
Figure 17. Greedy Search Result



```
answer=predict_answer('what are the signs of heart attack?')
print("answer:",answer)

answer: heart attacks are often caused by a combination of several factors including a viral infection or an underlying heart disease the most common ca
use of death is the inability of blood vessels to move in and out of your heart vessels in addition to being unable to breathe or move your arms or legs
it can also be a result of an enlarged heart or a heart defect in one or both of those vessels there is no way to diagnose or treat the disease in any w
ay other than to see your medical provider for an examination
```

Figure 18. Beam Search with ngram = 3 Result



```
In [ ]: answer=predict_answer('what are the alzheimer disease treatment')
        print(answer)

        Output:
        ----------------------------------------------------------------------------------------------------
        0: `ANSWER: alzheimer is disease is an extremely rare disease diagnosed with a combination of drugs that are both effective and
        preventative of disease and are associated with a lower risk population of being affected by other diseases the treatment usual
        ly involves using steroids and a prescription medication...

        1: `ANSWER: there are many different treatments available to treat your symptoms and symptoms of alzheimer disease treatment is
        not always the most effective treatment because many patients with alzheimer disease often have symptoms similar to the symptom
        s of the disease and often do not respond to...

        2: `ANSWER: it depends on the severity of the disease and whether the treatment is given to your particular type of disease or
        to other people in the same condition i recommend a biopsy for a blood test and for the blood test if possible if you are unsur
        e about...
```

Figure 19. TopK Sampling Result

Summary of Results:

Table 1. Experiment Results

Experiment	**Model**	**Training Time**	**Training Accuracy**	**Validation Accuracy**	**Training Loss**	**Validation Loss**
1	LSTM Seq2Seq	30 mins	23.38%	5.09%	0.52	1.98
2	Transformer	3 hrs	9.6%	8.1%	0.4	0.6
3	BioBert and GPT2	22 hrs	88.9%	78.%	0.4	0.6

Deployment

We successfully deployed our application with the majority of our features into our production environment. In this case, our production environment is a publicly accessible web application hosted by Azure App Services. Figure 20 depicts the deployed architecture of our application.

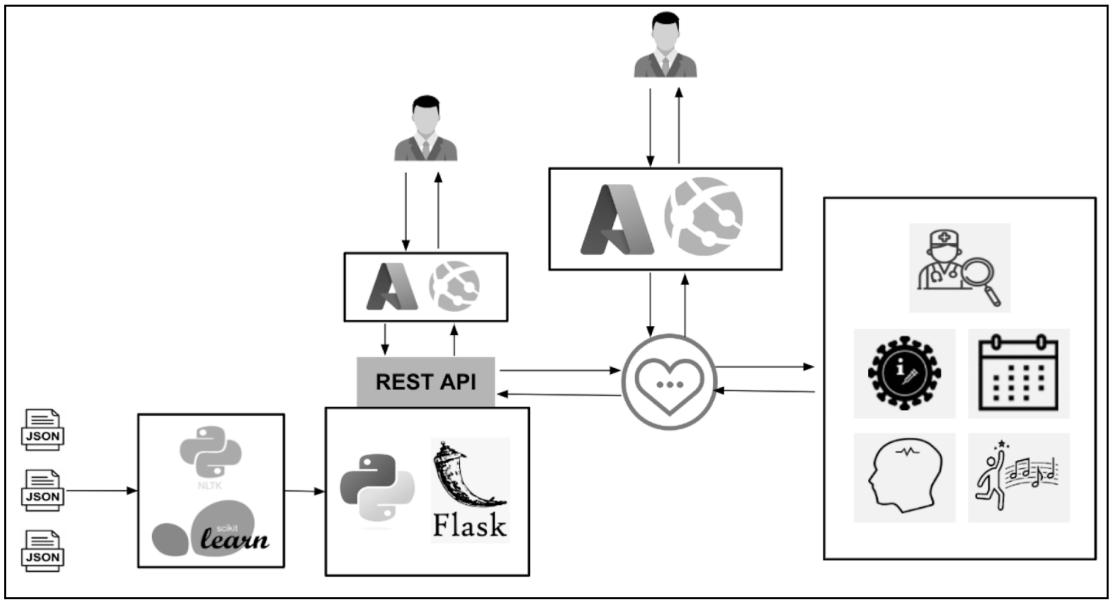
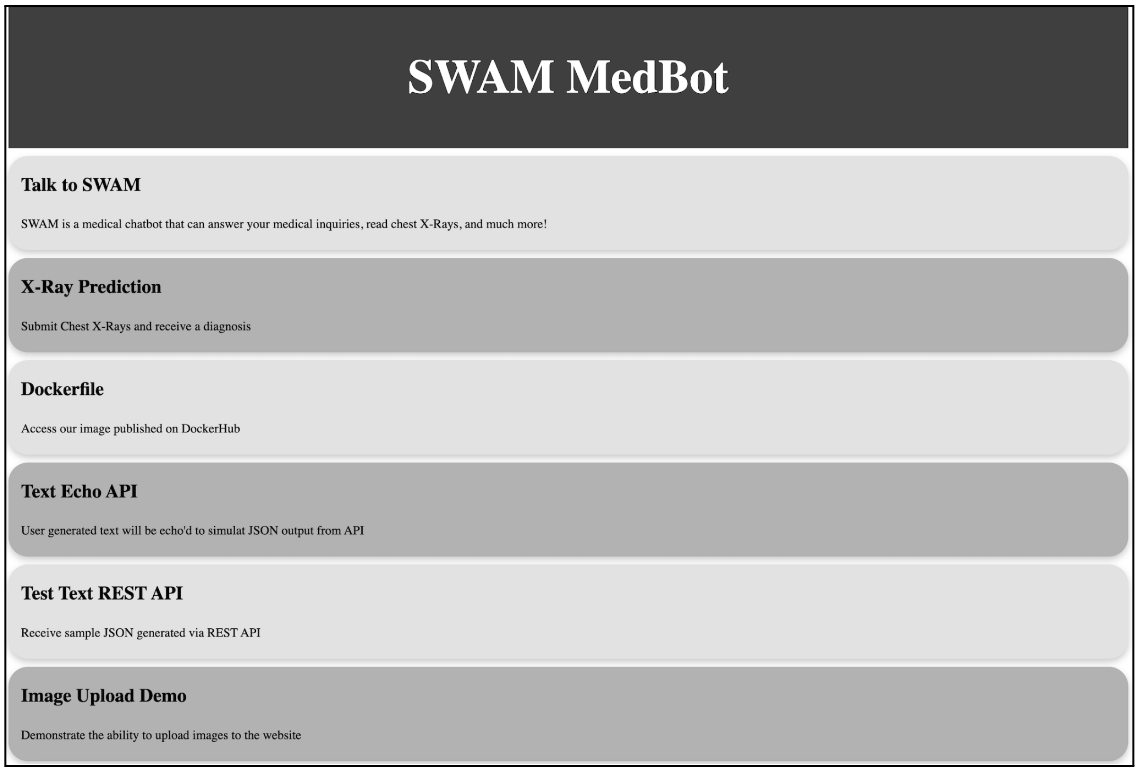


Figure 20. Production deployment architecture for SWAM medical chatbot application.

The first notable part of our deployed app consists of a user interface clients can use to interact with our image recognition model and the RESTful functionality of the application. SWAM UI is the name used to reference this portion of our architecture. SWAM UI is reachable at https://healthbot498.azurewebsites.net/. Figure 21 shows this UI developed for the left client interface depicted in figure 20.



SWAM MedBot

Talk to SWAM

SWAM is a medical chatbot that can answer your medical inquiries, read chest X-Rays, and much more!

X-Ray Prediction

Submit Chest X-Rays and receive a diagnosis

Dockerfile

Access our image published on DockerHub

Text Echo API

User generated text will be echo'd to simulat JSON output from API

Test Text REST API

Receive sample JSON generated via REST API

Image Upload Demo

Demonstrate the ability to upload images to the website

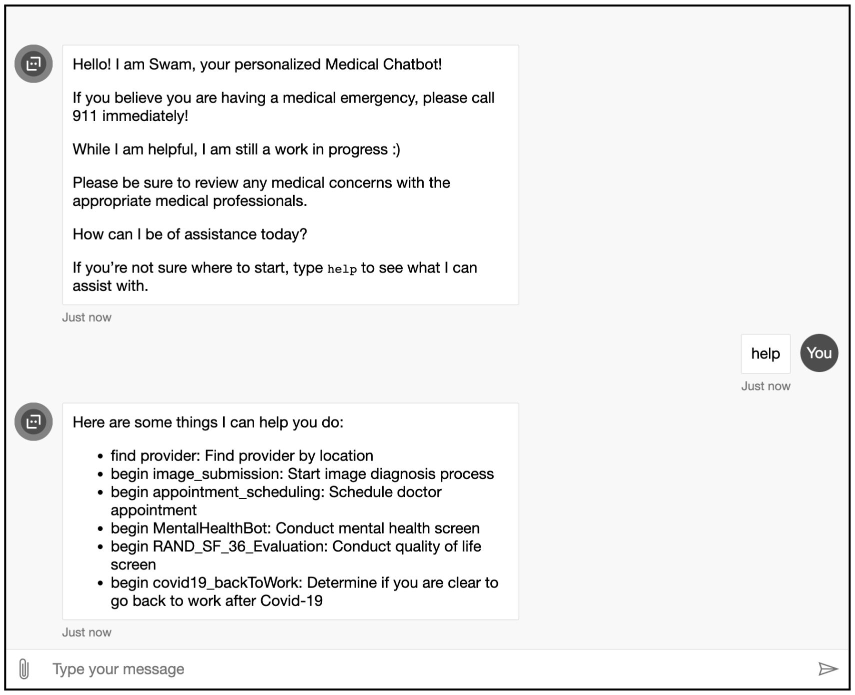
Figure 21. User interface developed for the application that is deployed into production and reachable at https://healthbot498.azurewebsites.net/.

The SWAM UI displays six cards clients can select to demonstrate the various functionality built into the application. For instance, the Text Echo API, Test Text REST API, and Image Upload Demo cards demonstrate the RESTful functionality of our application while processing both text and image inputs. The SWAM UI requires RESTful functionality to serve as a third-party API to the Azure Health Bot. In theory, this enables Azure Health Bot to submit text and image input to our models and receive a response.

The Dockerfile card will redirect clients to a published image of our application on DockerHub. A Docker image of our application is available at mikesoukup/msds498-chatbot. We chose to publish an image of our application on DockerHub to make our application code portable. In addition, this will facilitate the distribution and deployment of our app as a containerized microservice.

The X-Ray Prediction card will direct clients to an image submission form. Here, clients can upload frontal and profile chest X-ray images and receive a diagnosis for any cardiac or pulmonary abnormalities from a previously developed artificial intelligence model trained on an NIH image dataset.

Lastly, the Talk to SWAM card will route users to https://msds498swam.azurewebsites.net/. This will connect users to the Azure Health Bot interface and allow users to interact with our application features in a digitally native way.



Hello! I am Swam, your personalized Medical Chatbot!

If you believe you are having a medical emergency, please call 911 immediately!

While I am helpful, I am still a work in progress :)

Please be sure to review any medical concerns with the appropriate medical professionals.

How can I be of assistance today?

If you're not sure where to start, type `help` to see what I can assist with.

Just now

help

Just now

Here are some things I can help you do:

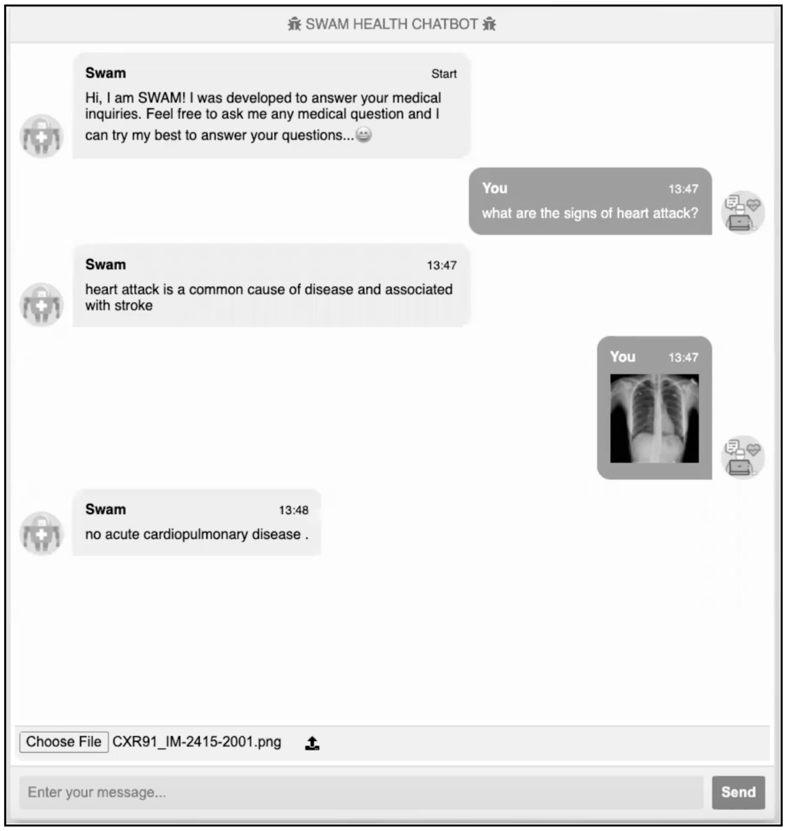
- find provider: Find provider by location
- begin image_submission: Start image diagnosis process
- begin appointment_scheduling: Schedule doctor appointment
- begin MentalHealthBot: Conduct mental health screen
- begin RAND_SF_36_Evaluation: Conduct quality of life screen
- begin covid19_backToWork: Determine if you are clear to go back to work after Covid-19

Just now

Figure 22. Azure Health Bot chatbot user interface located at https://msds498swam.azurewebsites.net/ with help menu displayed.

The Azure Health Bot interface provides a chat dialogue clients can use for self-service medical inquiries. Clients can ask a medical question directly into the textbox, or they can run several scenarios to extend the application's functionality. The help box displayed shows six scenarios a user could run that we have incorporated into our application. Entering begin image_submission, for example, will prompt the user to send two chest X-ray images to the image recognition model discussed earlier. This is the same model users could interact with by selecting the X-Ray Prediction card from the SWAM UI. In addition, the Azure Health Bot contained additional functionality, such as provider lookup and appointment scheduling. Users can engage with these features by typing `find provider` and `begin appointment_scheduling`, respectively. At this stage, the functionality for both of these features is only a demonstration. In order to fully enable these features, we must integrate our application with a medical practice's provider network and scheduling software. Lastly, we extended the functionality of our application by deploying three other health bot scenarios: a mental health screener (MentalHealthBot), a quality of life screener (RAND_SF_36_Evaluation), and a covid-19 back to work evaluation (covid19_backToWork).

Azure App Services and Azure Health Bot enabled us to deploy the application and functionality discussed above. However, despite our successes, we could not deploy the core functionality of our MVP, The BioBert GPT-2 Medical question and answer model, due to limitations of the Azure platform discussed in the Analysis and Interpretation section. As a result, we had to develop our own chat interface for users to interact with this model. This interface is similar to that of the Azure Health Bot interface. It displays a conversation and dialogue box for the client to ask medical questions to receive a response. In addition, the user can also upload X-Ray images to receive a diagnosis from our image recognition model. Figure 23 shows an image of this interface.



Swam Start
Hi, I am SWAM! I was developed to answer your medical inquiries. Feel free to ask me any medical question and I can try my best to answer your questions...

You 13:47
what are the signs of heart attack?

Swam 13:47
heart attack is a common cause of disease and associated with stroke

You 13:47

Swam 13:48
no acute cardiopulmonary disease .

Figure 23. Flask application interface for users to interact with the BioBert GPT-2 language model and our image recognition X-Ray prediction model.

Analysis and Interpretation

Experiment 1 (LSTM seq2seq):

We trained our encoder-decoder LSTM model over 1000 epochs for about 30 minutes and observed the training accuracy increasing per epoch. However, the validation accuracy is low and keeps fluctuating quite a bit. The fact that the loss kept rapidly increasing while accuracy fluctuated indicates some over-fitting. This approach for creating chatbots will need us to use a very large dataset to generate better responses to the user, as we observe that the chatbot generates some predictions. However, it does not generate coherent and accurate responses consistently. The current encoder-decoder model, although easier to train (the training time being much lower than the transformer models), lower in memory consumption, and simpler to understand, did not capture all the dependencies in the decoder layer due to the compact nature of LSTM. Hence, this was not our model of choice for the final product.

Experiment 2 (Transformer):

Looking at the training metrics through the 100 epochs for the transformer model in figure 13, we can see that the model's accuracy increases. This indicates that the model did not overfit. Another observation from the learning graph is that the validation accuracy plateaus did not increase significantly toward the end of the training.

The predictions from the transformer model are interpretable and coherent with relatively similar lengths, between 50 to 70 words. In addition, the predictions consistently start with greeting words such as “hi” and end with phrases such as “thank you” and “hope this helps you good luck” because of the positional embeddings in the transformer model. However, the responses generated by the Transformer model are unreliable after comparing it with several common medical questions and answers.

Experiment 3 (BioBERT and GPT-2):

BioBERT and GPT2 produced more accurate results compared to our traditional Transformer Model. While the model learning curve showed some signs of overfitting, we could resolve this by increasing the sample size. In the future, we will scrape more questions and answers from health websites to reduce overfitting so the model can generalize well. We used automated metrics to evaluate the generated responses called BiLingual Evaluation Understudy or Bleu Score. Blue Scores range from 0 to 1. Overall, we were able to generate decent-looking answers to most of the questions. However, Based on our observation, the Bleu score is high for some of the answers, whereas, in others, it is low. As shown in our results section, figure 16, 0.07 is a relatively low Bleu Score. One disadvantage of using automated metrics like the Bleu score for evaluation is that it fails to consider the meanings of words and focuses only on the exact match of words. Even two humans would likely come up with different sentence variants for a question and would rarely achieve a perfect match. For this reason, a score closer to 1 is unrealistic in practice and should raise a flag that the model is overfitting. As a result, we cannot judge our model based on this bleu score. The results generated by the model must be open-sourced to be assessed by clinical experts in order for us to evaluate it accurately.

Moreover, we also noticed some variations in responses in the GPT-2 model. For example, greedy search generates a concise and precise answer to a question but sometimes repeats itself. On the other hand, beam search and Top K sampling produce long sentences. Overall, top-K sampling seems to produce more fluent text than traditional greedy and beam searches on open-ended language generation. This is because in top-K sampling, only the K most likely next words are filtered, and the probability mass is redistributed among them. One of the reasons for GPT-2's success in story generation was its adoption of this sampling scheme.

The deployment of our models and applications experienced many successes and many challenges. We will start by highlighting the accomplishments of the work conducted so far. After our discussion on what went well, we will shift focus and review the challenges and shortcomings of our current deployed architecture.

As one of our primary goals, we wanted to make our minimum viable product publicly accessible so that anyone connected to the internet could interact with our models and obtain medical information. In this regard, we were successful. One can access our Azure App Services-hosted application at https://healthbot498.azurewebsites.net/ with a 99.95% uptime guarantee. At this web address, users can receive answers to their medical queries from the Azure Health Bot or receive cardiopulmonary screens from our deployed image recognition model. In addition, this web application enables users to access several features of our application in a digitally native way.

The application we deployed is simple, responsive, and RESTful. The RESTful utility is important as it enhances the user experience by enabling the functionality we built to interact with various other applications, such as the Azure Health Bot. The Azure Health Bot has ample rule-based scenarios developers can incorporate into a client interaction. This enables our application to provide a wide variety of user requests. The ease in creating, modifying, and deploying these scenarios was a highlight in the development process as it enabled us to easily incorporate features such as provider lookup, appointment scheduling, x-ray diagnosis, and mental health screens among various others.

Another win for our deployed application was that our architecture was modular and cloud supported. There are two ways to address the modularity of our application. The first thing we did was organize each feature of our application into a unique function. This ensures that clients who wish to simply receive an answer to their medical questions do not have to upload chest X-ray images. Similarly, one does not have to conduct a mental health screen during the provider lookup process. The application’s features and rule-based scenarios are siloed and independent, supporting a natural user experience. Second, our Flask-based application is a third-party API to the Azure Health Bot.

Consequently, we can develop our Flask app asynchronously while developing Azure Health Bot. Since the Flask app is REST-compliant, we can treat these two components of our architecture separately so long as those components exchange information in a REST-compliant manner. Furthermore, this same philosophy can provide opportunities to extend the Flask app's feature set or the entire system's feature set through other microservices. Additionally, the cloud-supported architecture amplifies the impact of this design flexibility. By doing this, we can easily adjust the resource requirements to meet the computational demands placed on our application when it is in production.

Our minimal viable product's final pillar of success is the chat interfaces we deployed. We deployed the chat interfaces to allow users to interact with our artificial intelligent models in a digitally native way. Users can ask questions, interact with scenarios, and upload images via an interface similar to chat interfaces most people use to send texts or interact with colleagues in today’s digital society. The importance of this aspect is that the models have gone beyond the development notebook environment and are now consumable to the entire public.

Now that we have discussed some highlights of the deployed architecture we designed, we will now discuss the challenges and roadblocks we faced. For instance, one major challenge we faced was our inability to deploy our BioBERT GPT-2 language model. Large file sizes and slow upload latency brought on this challenge. The three BioBERT GPT-2 dependencies needed to execute our language model consisted of the question_extractor_model, the medical_gpt2_model_new_2_107, and the train_gpt_data.pkl, which required 422 MB, 475 MB, and 266 MB disk space, respectively. In short, our AI BioBERT GPT-2 model required 1.1 GB of disk space alone to answer user queries. The challenge we faced was that when uploading these files via Azure App services, the model files would upload and appear in the virtual machine but not until after the startup script initiated. As such, the startup process would return File Not Found errors and crash the virtual machine hosting the application. Even after working with Azure Technical Support, we are still seeking a proper solution to this issue. However, one possible solution to this problem is storing these large language model files in a cloud storage solution such as Azure File Sync or Data Lake Storage Gen1. By doing this, the virtual machine is not ephemerally storing our model files. This means we do not have to upload large model files each time we restart our application. Instead, our application can locate these files in cloud storage, ensuring they are always present during application startup.

The second challenge we faced was the rigid HTTP POST timeout configuration associated with the data connection control element within Azure Health Bot. The HTTP POST timeout configuration is only set to ten seconds. This means if the control element does not get a response from the server it is reaching out to, in this case, our Flask API, the server connection will close, and the control element will raise an error. The issue arises as our x-ray recognition model has an average response latency of fifteen seconds while our BioBERT GPT-2 language has an average response latency of sixty seconds. As a result, the Azure Health Bot could send information such as images or text to our Flask API as part of the HTTP POST request, but the server connection would close before receiving a response rendering the Azure Health Bot interface useless for our application. Working with Azure Cloud Support revealed that the HTTP POST timeout request for the Azure Health Bot Data Connection control elements is not configurable, limiting this solution's compatibility with our application. This roadblock led to the migration from the Azure Health Bot to a newly developed solution. This interface, shown in Figure 23, was our remedy to this issue. We created a chatbot user interface using JavaScript configurable to our models' latency requirements. The drawback of this solution is that we are unable to incorporate predefined Azure Health Bot features such as provider lookup, appointment scheduling, or mental health screens. Moving forward, we require a solution that blends the features of both the JavaScript-developed chatbot UI and Azure Health Bot functionality for a harmonized and complete solution to maximize the user experience.

Conclusion

In this work, we advocate for BioBERT and GPT-2 model solutions to generate responses to user-provided medical questions. Our work demonstrated how to adopt BioBERT for encoding medical questions and answers to develop robust vector representation, use FAISS search for similarity lookup, and input those embeddings to the GPT-2 model for generating responses to medical questions. Our work reveals that BioBERT has innate capabilities to encode medical questions and answers. Jointly with GPT-2, the solution can generate accurate biomedical text responses. Our work proves a BioBERT and GPT-2 solution achieves better results than LSTM Seq2Seq and Transformer encoder-decoder models for end-to-end question-answering tasks. It also demonstrates better biomedical text generation ability than GPT-2 on the text generation task. Lastly, our work illustrates how to deploy these models via a solution architecture using the Flask, Azure Health bot, and Azure APP service. Through the developed chat interfaces, patients can ask medical questions, submit X-Ray images for the diagnosis, and even schedule medical appointments. This MVP can potentially revolutionize healthcare by connecting more people to prompt, high-quality healthcare.

Directions for Future Work

Moving forward, our focus will be on two main objectives: build out our feature set beyond a working MVP and productionalize our product. One way to take our product beyond an MVP is to improve our language model to return more natural responses. The objectives here are to ensure the responses delivered from our language model are accurate and can pass a Turing test. To achieve this, we will increase the sample size of our dataset and sift through the data to eliminate any biases in responses. Additionally, we will explore pre-trained models such as BioGPT from Microsoft, which can serve as the hybrid power of BioBERT and GPT-2. We believe this direction will lead to success as BioGPT uses GPT-2 as its primary model, has achieved human parity, and outperformed other general and scientific LLMs (Luo et al. 2022b). In addition, BioGPT has been trained on 15 million PubMed abstracts, making it easier to generate an accurate and natural response. Second, we need to either re-develop our chat interface with alluring features such as provider lookup and appointment scheduling or find another out-of-the-box solution that is compatible with our Flask API. Doing so will ensure our medical chatbot is robust and can satisfy a high percentage of users. Third, we must make our application FHIR compliant to facilitate the exchange of personally identifiable medical information and protect the storage of the data we collect on users. Once these developments are complete, the last step to productionalize our product will be to align with a medical practice to integrate the solution we developed with their provider networks, scheduling software, and client base so users can benefit from what we have created.

References

———. 2022b. “BioGPT: Generative Pre-Trained Transformer for Biomedical Text Generation and Mining.” *Briefings in Bioinformatics*, September. https://doi.org/10.1093/bib/bbac409.

Adamopoulou, Eleni, and Lefteris Moussiades. 2020. “Chatbots: History, Technology, and Applications.” *Machine Learning with Applications* 2 (December). https://doi.org/10.1016/j.mlwa.2020.100006.

Bhirud, Nivedita, Subhash Tataale, Sayali Randive, and Shubham Nahar. "A literature review on chatbots in healthcare domain." International journal of scientific & technology research 8, no. 7 (2019): 225-231.

Caldarini, Guendalina, Sardar Jaf, and Kenneth McGarry. 2022. “A Literature Survey of Recent Advances in Chatbots.” *Information* 13 (1): 41. https://doi.org/10.3390/info13010041.

California Consumer Privacy Act of 2018. Cal. Civ. Code §§ 1798.100-1798.199 (2018)

Devlin, Jacob, Chang, Ming, Lee, Kenton, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *ArXiv*, (2018). Accessed February 5, 2023. https://doi.org/10.48550/arXiv.1810.04805.

“Doc Product: Medical Q&a with Deep Language Models.” n.d. Devpost. Accessed February 4, 2023. https://devpost.com/software/nlp-doctor.

Health Insurance Portability and Accountability Act of 1996. Pub. L. No. 104-191, 110 Stat. 1936 (1996).

Ibrihich, S., A. Oussous, O. Ibrihich, and M. Esghir. 2022. “A Review on Recent Research in Information Retrieval.” *Procedia Computer Science* 201 (January): 777–82. https://doi.org/10.1016/j.procs.2022.03.106.

"Indexing 1G Vectors." n.d. GitHub. Accessed February 5, 2023. https://github.com/facebookresearch/faiss/wiki/Indexing-1G-vectors.

Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019a. "BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining." Edited by Jonathan Wren. *Bioinformatics* 36 (4). https://doi.org/10.1093/bioinformatics/btz682.

Li, Bryan M. 2019. "A Transformer Chatbot Tutorial with TensorFlow 2.0." TensorFlow Blog. May 23, 2019. https://blog.tensorflow.org/2019/05/transformer-chatbot-tutorial-with-tensorflow-2.html.

Lommatzsch, Andreas, and Jonas Katins. 2019. "An Information Retrieval-Based Approach for Building Intuitive Chatbots for Large Knowledge Bases." Semantic Scholar. 2019. https://www.semanticscholar.org/paper/An-Information-Retrieval-based-Approach-for-for-Lommatzsch-Katins/cc07ada9c2356bb6d19afba797cb5a7a50a0e0d7.

Mauldin, Michael L. "Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. "In AAAI, vol. 94, pp. 16-21. 1994.

Martin, Saleen. 2022. "More Americans Have Gotten Mental Health Treatment since 2019, Especially Younger Adults and Women." USA TODAY. September 18, 2022. https://www.usatoday.com/story/news/health/2022/09/18/more-adults-received-mental-health-treatment-over-past-two-years/10369715002/.

Microsoft Azure. n.d. "Azure Health Bot | Microsoft Azure." Azure.microsoft.com. Accessed February 4, 2023. https://azure.microsoft.com/en-us/products/bot-services/health-bot.

Nielsen, Lasse Regin. 2017. "GitHub - LasseRegin/Medical-Question-Answer-Data: Medical Question and Answer Dataset Gathered from the Web." GitHub- Medical Question and Answer Dataset Gathered from the Web. May 5, 2017. https://github.com/LasseRegin/medical-question-answer-data.

Nirala, Krishna Kumar, Nikhil Kumar Singh, and Vinay Shivshanker Purani. 2022. "A Survey on Providing Customer and Public Administration Based Services Using AI: Chatbot." *Multimedia Tools and Applications*, January. https://doi.org/10.1007/s11042-021-11458-y.

NRC Health. 2019. "2019 Healthcare Consumer Trends Report." NRC Health. https://go.nrchealth.com/l/279972/2018-12-06/3vnpd.

Oniani, David, and Yanshan Wang. 2020. "A Qualitative Evaluation of Language Models on Automatic Question-Answering for COVID-19." *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, no. 33 (September): 1–9. https://doi.org/10.1145/3388440.3412413.

Osborn, Robin, David Squires, Michelle M. Doty, Dana O. Sarnak, and Eric C. Schneider. 2016. "U.S. Adults Still Struggle with Access to and Affordability of Health Care." Www.commonwealthfund.org. November 16, 2016. https://www.commonwealthfund.org/publications/journal-article/2016/nov/new-survey-11-countries-us-adults-still-struggle-access-and.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. "Language Models Are Unsupervised Multitask Learners." https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Rakib, Afsana Binte, Esika Arifin Rumky, Ananna J. Ashraf, Md. Monsur Hillas, and Muhammad Arifur Rahman. 2021. "Mental Healthcare Chatbot Using Sequence-To-Sequence Learning and BiLSTM." *Brain Informatics*, 378–87. https://doi.org/10.1007/978-3-030-86993-9_34.

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). OJ L 119, 4.5.2016, p. 1.

Shah, Dhruvil. 2020. “Generative Chatbots Using the Seq2seq Model!” Medium. July 28, 2020. https://towardsdatascience.com/generative-chatbots-using-the-seq2seq-model-d411c8738ab5.

Shum, Heung-yeung, Xiao-dong He, and Di Li. 2018. “From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots.” *Frontiers of Information Technology & Electronic Engineering* 19 (1): 10–26. https://doi.org/10.1631/fitee.1700826.

Substance Abuse and Mental Health Services Administration (SAMHSA). N.d. “SAMHSA’s Mission.” SAMHSA.gov. https://www.samhsa.gov/about-us/samhsas-mission.

Sutskever, Ilya, Vinyals, Oriol, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks." ArXiv, (2014). Accessed February 5, 2023. https://doi.org/10.48550/arXiv.1409.3215.

Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. “Efficient Transformers: A Survey.” *ACM Computing Surveys* 55 (6). https://doi.org/10.1145/3530811.

Turing, A. M. 1950. “Computing Machinery and Intelligence.” *Mind* LIX (236): 433–60. https://doi.org/10.1093/mind/lix.236.433.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Gomez, Aidan N, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” ArXiv.org. June 12, 2017. https://arxiv.org/abs/1706.03762.

Vinyals, Oriol, and Quoc Le. "A Neural Conversational Model." *ArXiv*, (2015). Accessed February 5, 2023.https://doi.org/10.48550/arXiv.1506.05869.

Wagner, Joshua. 2020. Chapter 8 Attention and Self-Attention for NLP | Modern Approaches in Natural Language Processing. Slds-Lmu.github.io. published with bookdown. https://slds-lmu.github.io/seminar_nlp_ss20/attention-and-self-attention-for-nlp.html.

Weizenbaum, Joseph. 1966. "ELIZA---a Computer Program for the Study of Natural Language Communication between Man and Machine." *Communications of the ACM* 9 (1): 36–45. https://doi.org/10.1145/365153.365168.

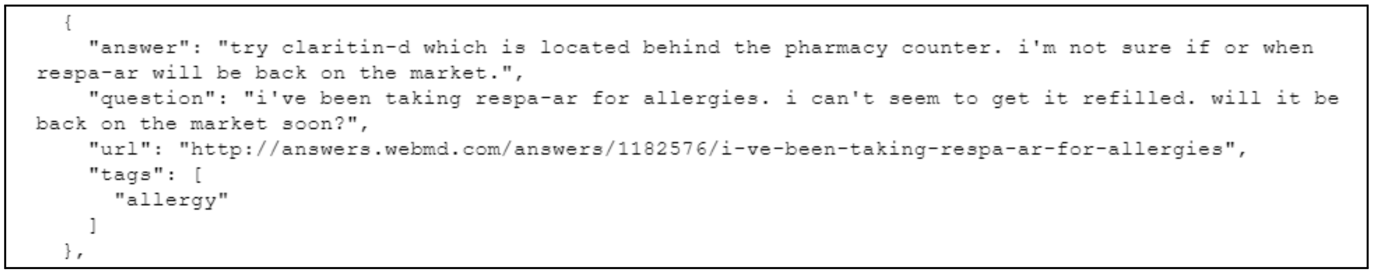
"Welcome to Flask — Flask Documentation (2.2.x)." n.d. Flask.palletsprojects.com. Accessed January 15, 2023. https://flask.palletsprojects.com/en/2.2.x/.

Xu, Anbang, Zhe Liu, Yufan Guo, Vibha Sinha, and Rama Akkiraju. 2017. "A New Chatbot for Customer Service on Social Media." *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. https://doi.org/10.1145/3025453.3025496.

Yang, Christopher C. 2022. "Explainable Artificial Intelligence for Predictive Modeling in Healthcare." *Journal of Healthcare Informatics Research* 6 (February). https://doi.org/10.1007/s41666-022-00114-1.

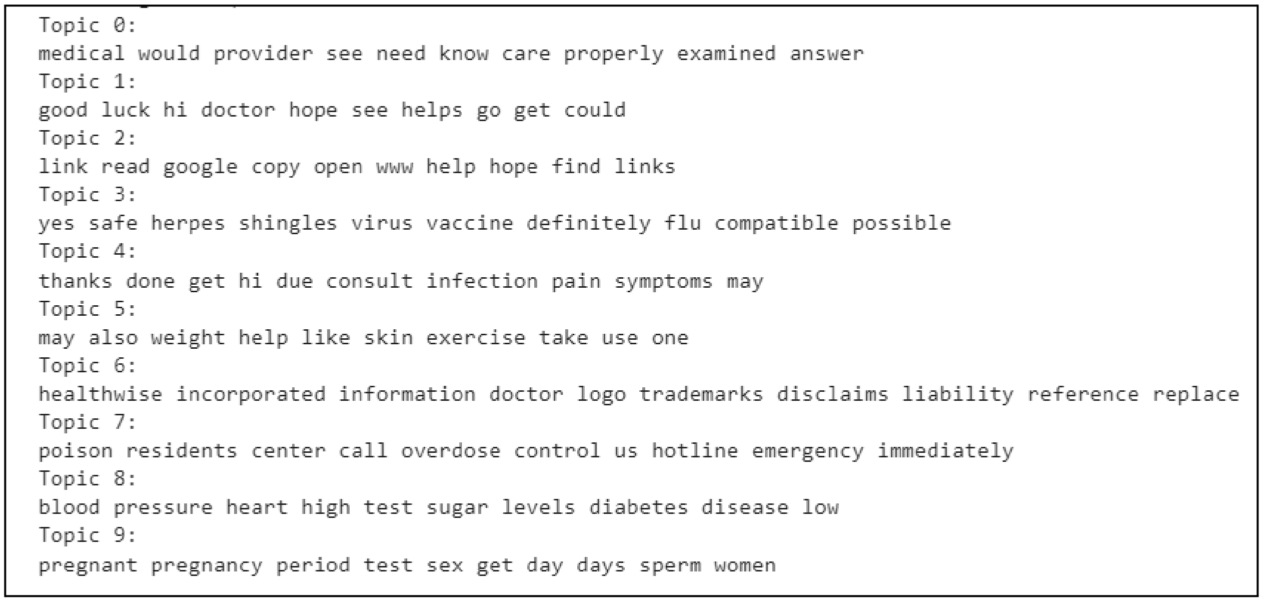
Zhou, Li, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. "The Design and Implementation of XiaoIce, an Empathetic Social Chatbot." *Computational Linguistics* 46 (1): 53–93. https://doi.org/10.1162/coli_a_00368.

Appendix A



```
{
  "answer": "try claritin-d which is located behind the pharmacy counter. i'm not sure if or when
respa-ar will be back on the market.",
  "question": "i've been taking respa-ar for allergies. i can't seem to get it refilled. will it be
back on the market soon?",
  "url": "http://answers.webmd.com/answers/1182576/i-ve-been-taking-respa-ar-for-allergies",
  "tags": [
    "allergy"
  ]
},
```

Figure A1. Example of the question and answer dataset



```
Topic 0:
medical would provider see need know care properly examined answer
Topic 1:
good luck hi doctor hope see helps go get could
Topic 2:
link read google copy open www help hope find links
Topic 3:
yes safe herpes shingles virus vaccine definitely flu compatible possible
Topic 4:
thanks done get hi due consult infection pain symptoms may
Topic 5:
may also weight help like skin exercise take use one
Topic 6:
healthwise incorporated information doctor logo trademarks disclaims liability reference replace
Topic 7:
poison residents center call overdose control us hotline emergency immediately
Topic 8:
blood pressure heart high test sugar levels diabetes disease low
Topic 9:
pregnant pregnancy period test sex get day days sperm women
```

Figure A2. Non-negative Matrix Factorization (NMF) Topic Modeling Results with TF-IDF