

PERSONALIZED HEALTH ASSISTANT SMART BOT USING ARTIFICIAL INTELLIGENCE

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in
Information Technology

By

**ABHINAV KUMAR (Reg No. 39120003)
SHAURYA SAMEER (Reg No. 39120110)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

MAY - 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Abhinav Kumar (Reg No. 39120003)** and **Shaurya Sameer (Reg No. 39120110)** who carried out the Project Phase-2 entitled "**PERSONALIZED HEALTH ASSISTANT SMART BOT USING ARTIFICIAL INTELLIGENCE**" under my supervision from January 2023 to May 2023.

Internal Guide

Dr. R. SUBHASHINI M.E., Ph.D.

Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.

Submitted for Viva voce Examination held on _____

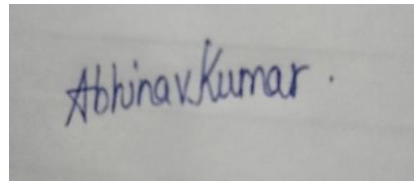
Internal Examiner

External Examiner

DECLARATION

I, **Abhinav Kumar (Reg No. 39120003)**, hereby declare that the Project Phase-2 Report entitled “**PERSONALIZED HEALTH ASSISTANT SMART BOT USING ARTIFICIAL INTELLIGENCE**” done by me under the guidance of **Dr. R. Subhashini, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award Bachelor of Technology degree in Information Technology.

DATE:

A rectangular box containing a handwritten signature in blue ink that reads "Abhinav Kumar .".

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing and **Dr. R. SUBHASHINI M.E., Ph.D.**, Head of the Department, Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. Subhashini, M.E., Ph.D.**, for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

The importance of maintaining health has been recognized around the world, but the number of professionals offering such services has been on a decline, and therefore not everyone can receive medical consultations on a regular basis. According to statistics, most people experienced heart disease and lung disease and at the same time noted that they had never visited a doctor. Most of the people are not medically qualified for studying or understanding the extremity of their diseases or symptoms. This is the place where natural language processing plays a vital role in healthcare. These chatbots collect patients' health data and depending on the data, these chatbot give more relevant data to patients regarding their body conditions and recommends further steps as well. In the medical field, AI powered healthcare chatbots are beneficial for assisting patients and guiding them in getting the most relevant assistance. Chatbots are more useful for online search that users or patients go through when patients want to know for their health symptoms. In this project, the health assistant system was developed using Bio-BERT algorithm which is a Google's Natural language processing powered algorithm that is fed medical data instead. With this web application, a user can make health requests/queries via text message and might also get relevant health suggestions/recommendations through it. The virtual assistant will take symptoms from user and then it analyses the percentage of associated disease and finds the nearest doctor to that location. This chatbot acts like an informative and conversational chatbot. In the healthcare sector AI-powered applications have seen a remarkable spike in recent days. The Covid crisis changed the whole healthcare system upside down. So, this NLP powered chatbot system can reduce office wait, save money, time and energy. Patients might be getting medical knowledge and assisting ourselves within their own time and place.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
	LIST OF TABLES	x
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Literature review	3
	2.2 Inferences from literature survey	8
	2.3 Open problems in existing system	9
3	REQUIREMENTS ANALYSIS	10
	3.1 Feasibility studies/Risk analysis of the project	10
	3.2 Software requirements specifications document	11
4	DESCRIPTION OF PROPOSED SYSTEM	12
	4.1 Selected methodology or process model	12
	4.2 System Architecture	17
	4.3 Description of software for implementation and testing plan of the proposed system	18
	4.4 Project management plan	21
5	IMPLEMENTATION DETAILS	23
	5.1 Development and deployment setup	23
	5.1.1 Business Problem Overview	23
	5.1.2 ML formulation of the problem	23
	5.1.3 Dataset analysis	24

5.2	Algorithms	25
5.2.1	Finetuning BioBert and Training Question Embedding Extractor	25
5.2.2	Fine tuning the GPT-2 model	30
5.3	Testing	34
6	RESULTS AND DISCUSSIONS	35
7	CONCLUSION	36
7.1	Conclusion	36
7.2	Future work	36
7.3	Research Issues	37
7.4	Implementation Issues	38
	REFERENCES	40
	APPENDIX	42
A.	SCREENSHOTS	42
B.	SOURCE CODE	44
C.	RESEARCH PAPER	52

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.2	System Architecture	22
5.1	Sample Data	29
5.2	Data Preparation for Question Embedding Extractor Model	30
5.3	BioBert epoch vs loss and accuracy	33
5.4	Comparison of cosine similarities of correctly classified negative and positive points	33
5.5	Comparison of cosine similarities of incorrectly classified negative and positive points	34
5.6	Extracting Question and Answer Embeddings	35
5.7	GPT-2 Data Pipeline	36
5.8	GPT-2 epoch vs loss	37
5.9	Training samples	38
5.10	Validation samples	38

LIST OF ABBREVIATIONS

S No.	ABBREVIATIONS	DEFINITION
1	NLP	NATURAL LANGUAGE PROCESSING
2	API	APPLICATION PROGRAMMING LANGUAGE
3	BERT	BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS
4	GPT-2	GENERATIVE PRE-TRAINED TRANSFORMER 2
5	UI	USER INTERFACE
6	JSON	JAVASCRIPT OBJECT NOTATION

CHAPTER 1

INTRODUCTION

People have been so focused on their jobs over the course of the last several decades that they have forgotten to make prioritizing their health a continuous part of their daily routine. This has resulted in a widespread epidemic of poor health. This pattern is still present today and is rapidly increasing. This predicament, in the end, puts an individual's quality of life in peril throughout the course of a lengthier period. Nevertheless, with the assistance of artificial intelligence, we are now able to provide people access to health care services at times and locations of their choosing, as well as at costs that are within their financial limits. One of the most important things to consider A healthy body is one of the most precious gifts that God has bestowed upon us. Everyone wants to improve their overall quality of life and have a healthy body in which to live. This is something that everyone of us strives to achieve. The key objective of this essay is to provide these services to successfully fulfil the goal that was outlined earlier in the introduction. It is difficult to imagine what it will be like for us to continue existing. sans sophisticated technological devices since we now rely on them in almost every aspect of our lives. As a result, the study of artificial intelligence is becoming an increasingly significant topic, accelerating its growth due to the several uses it may serve in the scientific sector's rising need for it. One of the key focuses of the work that scientists do is illness prediction, which receives a large amount of attention and consideration. The results of the analysis of large data sets in turn enhance the accuracy of risk categorization based on the findings of a considerable quantity of data volume. In general, e-healthcare facilities are an essential resource for low-income nations, but, due to a lack of infrastructure, it may be challenging to create these facilities at times, mainly since there is a deficiency in both the gathering of information and the construction of infrastructure. Most people who use the internet rely primarily on this platform as their primary means of clearing their questions pertaining to medical care and treatment. We have developed a platform that can offer patients medical services in the form of online consultations, and our ultimate objective is to provide patients access to this platform so that they may take advantage of these consultations. This will help those who have found

employment in the healthcare sector of the economy. In addition, the user is given access to a more convenient method of obtaining medical advice as well as expanded opportunities to choose from a wider range of available illnesses and any tests that are now accessible for treating those disorders. To improve the efficiency of communication and the flow of the process as a whole, our system has been updated to include a chatbot. A diagnosis has been made on the patient's condition. Chatbots are pieces of software that simulate human conversation by using artificial intelligence (AI) and natural language processing (NLP). AI drives the chatbots, while NLP is used to translate what the chatbots say, understand, and then, based on that understanding, offer a response to the user that takes into account what was just interpreted. The outcomes of this research, which incorporates NLP and AI ideas, are discussed in terms of their prospective applications, one of which being the creation of a chatbot that can diagnose illnesses.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

[1] L. Athota, V. K. Shukla, N. Pandey and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020.

Healthcare is very important to lead a good life. However, it is very difficult to obtain the consultation with the doctor for every health problem. The idea is to create a medical chatbot using Artificial Intelligence that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. The chatbots are computer programs that use natural language to interact with users. The chatbot stores the data in the database to identify the sentence keywords and to make a query decision and answer the question. Ranking and sentence similarity calculation is performed using n-gram, TFIDF and cosine similarity. The score will be obtained for each sentence from the given input sentence and more similar sentences will be obtained for the query given. The third party, the expert program, handles the question presented to the bot that is not understood or is not present in the database.

[2] P. Anandan, S. Kokila, S. Elango, P. Gopinath and P. Sudarsan, "Artificial Intelligence based Chat Bot for Patient Health Care," 2022 International Conference on Computer Communication and Informatics (ICCCI), 2022.

India is a country with an increasing population of more than 1.3 billion, yet lack of medication is a crisis here. While travelling around government hospitals and as well as public hospitals, it is discovered that there are insufficient numbers of doctors to handle the growing population, where lack of sufficient doctors are the one of the reasons for patient deaths. To overcome these issues, we set to create a medical chatbot using Artificial Intelligence, which has the potential to provide an immediate access about any medical information-based queries. These chatbots

helps the patient to discover the nearby specialists and fixing their appointments. If the user is suffering from severe fever by asking few questions in series the chatbot provides the precautions based on the users' answers or it will suggest any specialists nearby according to the timeline of both user and specialist. In case if the user is searching for any diagnosis centre, for example if he/she needs any MRI scan the chatbot provides the information of labs in surroundings along with the opening and closing times. The design of the software is done through the combination of NLP and Machine Learning.

[3] M. G, "Mega Bot – The Healthcare Chatbot," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021.

This project aims to help people to check their health condition before going to a hospital this saves a lot of time and value and also allows you to know your symptoms in advance. It can perform tasks related to symptoms same like a doctor who takes a checkup but in a more accurate way. Machine Learning (ML), Artificial Intelligence (AI), Deep-Learning (DL) and IOT are used to detect the symptoms and checks the temperature. This algorithm using dependencies such as pysklearn, nltk, Tensor Flow, OpenCV, AI, and Raspberry Pi. It usually enters by entering the login details and then it asks you to click the start button and then you can say "yes" or "no" and then it finally uses to suggest the online doctors. There are also way more features like face checkup, fever checkup and third party infermedica API. The face checkup is used to check the emotions and fever is used to detect temperature and then third party infermedica API deals with your symptoms.

[4] R. Goel, R. P. Goswami, S. Totlani, P. Arora, R. Bansal and D. Vij, "Machine Learning Based Healthcare Chatbot," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022.

Health care is very significant for you to live a well and healthy life. Often people do not know all the treatments or symptoms associated with a particular disease. For young people, users with the problem go to the hospital in person for a laborious test and it is very problematic to get a doctor's appointment for any

health problems and managing phone calls. These types of problems can be solved by using Healthcare/Medicinal Chatbot by providing appropriate guidance regarding healthy living. The notion is to create a medical chatbot using Neural Networks that can provide with the info and diagnose the disease and deliver basic information about the disease and when and where to consult a doctor. The effectiveness of medical chatbots rest on Natural language processing technique that helps users post their concerns about any disease and also their health. The user can ask any private question related to health care via chatbot without being actually present at the clinic or hospital. This will help reduce the cost of health care and improve access to medical information through medical chat-bot. The program development plan is to analyze customer feelings.

[5] P. I. Prayitno, R. P. Pujo Leksono, F. Chai, R. Aldy and W. Budiharto, "Health Chatbot Using Natural Language Processing for Disease Prediction and Treatment," 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), 2021.

People who don't know about products or services provided by a company need a system that can provide answers to the questions that are usually asked. This system is called Frequently Asked Questions. This system is not effective and efficient in providing the information. In this paper, we propose a chatbot using Natural Language Processing to provide some information about health. Here, we use NLP to make a chatbot system that can understand and answer the questions that are asked by the user. The cosine similarity is used to find the similarities between the query words (the questions that are asked by the user) and the documents, then return the answers of the document with the highest similarity. Based on our development, our medical chatbot has successfully diagnose the user illness with approximately 87 percent accuracy.

[6] S. Badlani, T. Aditya, M. Dave, and S. Chaudhari, "Multilingual Healthcare Chatbot Using Machine Learning," 2021 2nd International Conference for Emerging Technology (INCET), 2021

The healthcare sector is one of the largest focus areas in the world today. Health problems are becoming increasingly common. India faces a huge challenge in terms of managing rural healthcare. Early diagnosis and treatment of diseases can

play an instrumental role. Physical consultation, particularly in the rural areas, is costly and time consuming. The solution is adopting healthcare chatbots. The proposed solution describes a multilingual healthcare chatbot application that can perform disease diagnosis based on user symptoms. It also responds to user queries by calculating sentence similarity by using TF-IDF and Cosine Similarity techniques and choosing the most appropriate response from its knowledge database. The multilingual capabilities of the chatbot system make it highly suitable for use in rural India. The chatbot application currently supports three languages namely English, Hindi and Gujarati. The chatbot application converses with the user using concepts of Natural Language Processing and also supports speech to text and text to speech conversion so that the user can also communicate using voice. Five different Machine Learning algorithms have been analyzed for disease prediction. The Random Forest Classifier produces the best results and gives an accuracy of 98.43%. Thus, it is used as the system's core classifier.

[7] J. Gupta, V. Singh and I. Kumar, "Florence- A Health Care Chatbot," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021.

The most common means by which unwell people receive health related analysis, disease diagnosis, and medicine prescription are hospitals. This has almost become the norm of all individuals around the world. Hospitals are considered the primary and the most reliable means of diagnosis. The proposed idea of this is to make it easier for people to check on their health as compared to the conventional way of standing in a queue for hours before they could get their medication done. To build a chatbot, this research aims to apply the use of the RASA framework. As any person, the chatbot can connect with people and take on the user's symptoms. It will then identify the most likely disease and predict it along with the treatment recommended. This will help people get a quick answer to all their queries without any hassle. This type of system is less known among the people and hence it is not widely used. It will also be of great use to record the nutrients the user has consumed throughout the day.

[8] P. Kandpal, K. Jasnani, R. Raut and S. Bhorge, "Contextual Chatbot for

Healthcare Purposes (using Deep Learning)," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020.

As the demand in Machine Learning & AI keeps growing, new technologies will keep coming in the market which will impact our day-to-day activities, and one such technology is Virtual Assistant Bots or simply Chatbots. Chatbots have evolved from being Menu/Button based, to Keywords based and now Contextual based. The most advanced among all of the above is contextual based because it uses Machine Learning and Artificial Intelligence techniques to store and process the training models which help the chatbot to give better and appropriate response when user asks domain specific questions to the bot. In this paper we will be not only discussing about the working of our model but also the applications and relevant work conducted in this domain, also there will be discussion about the challenges and future scope of this technology. For this work, neural networks have been used to train data and various packages which help us in giving better results. In this chatbot we will be integrating the concepts of Natural Language Processing with Deep Learning for getting better results. Healthcare plays a wide role in our daily lives, whenever a person feels sick, he/she visits their family doctor or any nearby clinic just to get to know what issues they are facing, in the recent years many companies & institutions have collaborated with hospitals to provide support which can help doctors and medical staff to deal with patients in better way and reduce their efforts with the help of technology. Chatbots can play a major role in reshaping the healthcare industry by providing either predictive diagnosis or any other assistantships like booking an appointment.

[9] S. Swain, S. Naik, A. Mhalsekar, H. Gaonkar, D. Kale and S. Aswale, "Healthcare Chatbot System: A Survey," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), 2022.

A chatbot is a software that produces a computer model of conversation through auditory or textual method between the user and the computer. Such bots use AI and NLP to understand user's questions and automate responses to them, simulating human conversation. In this modern technology almost, every person utilize smartphones in their day to day lives. Simply because, it has made life of a person simple by the touch of the screen, from extracting information from the internet, to provide personal assistance to entertainment everything happens just

by the click of a button. Today Chatbots are used in various domains like education, customer service, medical, website help etc for seeking information, FAQ answering and guidance. Due to the advancements in AI and techniques used in data mining, healthcare chatbots are often used for medical screening. The aim of this paper is to do detailed study of various research papers related to chatbots and analyze the different tools, algorithms, software, and platforms that are utilized.

[10] S. P. Konapur, T. Krishna, V. G, U. R and S. H, "Design of a Chatbot for People Under Distress using Transformer Model," 2021 2nd Global Conference for Advancement in Technology (GCAT), 2021.

As indicated by the WHO, the variety of mental issues incorporates dementia, discouragement, schizophrenia and other psychoses, intellectual disabilities, and developmental disorders. Sadness alone influences more than 300 million individuals. Indeed, even with admittance to proficient medical care and social administrations, those tormented with emotional well-being conditions wonder whether to profit from the therapy because of the stigma encompassing the disease. Accordingly, to this we would like to build an artificial intelligent agent that responds and helps the patient using an interactive chat bot. With the recent changes in Deep learning technology and specifically in NLP with models such as BERT and Open-AI GPT getting efficient contextual responses is not a task too hard. The chatbot will be able to identify, address and help relieve the mental stress of the user through soothing message responses like a therapist's conversation. Also, other relaxation techniques are provided on the same platform for further support to the distressed user.

2.2 INFERENCES FROM LITREATURE SURVEY

From the above-mentioned literature works, there has been effective research on this topic has been done and many models have been proposed. It is evident that the above-mentioned systems have their own pros and cons. While some of the recent works involve hybrid technologies and provide better accuracies, they are still far from what is needed. With higher accuracy, comes the need for low computational costs, high processing speed, and most of all, the convenience of

use.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system are not done with real time dataset and the existing system has low accuracy and low efficiency in terms of loading time and implementation time. Also, the testing and training is not done with the proper test-train split ratio.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

FEASIBILITY STUDY

The feasibility of the project is server performance increase in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the

user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications:

- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above, preferably Pentium or above

Software specifications:

- Python 3.6 and higher
- Anaconda and necessary libraries (Pytorch, Transformers, streamlit etc)

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Module 1 - Data collection and preprocessing

In this module, we start by making data more meaningful and informative is the effort of changing it from a given form to one that is considerably more useable and desired. This entire process can be automated using Machine Learning algorithms, mathematical modelling, and statistical expertise. Depending on the task we are conducting and the needs of the machine, the output of this entire process can be in any desired form, including graphs, movies, charts, tables, photos, and many more.

Getting the data and making it ready

The dataset must contain the necessary text prompts to build a complete chatbot. Using the Kaggle platform, which also received some of its questions from Google, the input texts, questions, and replies were gathered.

The project's prototype system was adapted from Kaggle to incorporate questions and replies from actual conversations into the model. The information is divided into many divisions, such as greetings, expressing gratitude by saying "thank you," replies relating to mental health, payment information, and many more. The obtained data will ultimately be used by the model in two modules, NLTK and NumPy.

Making of the dataset

- Created data for the chatbot like real life question answer to feed the model and train based on it.
- We created JSON format data for this case.
- The data is created in 3 segments 1 tags (for defining the class of query or statement user puts as input).
- We have created data to identify the input in different classes, the classes are greetings, goodbye, thanks, mental, payments, skin, fracture, delivery,

funny.

Sample data:

```
{
  "Tag": "greeting",
  "Patterns": [
    "Hi",
    "Hey",
    "How are you",
    "Is anyone there?"
  ]
  "Responses": [
    "Hey :-), I am Villy nice to see you.",
    "Hello thanks for visiting.",
    "Hi there, what can I do for you?"
  ]
}
```

Cleaning the data and making it ready for training

Data cleaning is the process of removing erroneous, damaged, badly structured, duplicate, or incomplete data from a dataset. When combining different data sources, we have checked this in a variety of methods to see if any data is duplicated or improperly categorized.

This module cleans the data by getting rid of duplication, hateful or irrelevant content, fixing structural issues, and dealing with missing data.

We eliminate observations with missing values as a last option, but keep in mind that doing so will remove or damage data.

We make up the missing figures based on other observations, but this approach again runs the risk of tainting the data's objectivity because you can be working with conjectures rather than concrete evidence.

As a third option, we can tweak how the data is utilized to efficiently navigate null

values.

Tokenization

Tokenization is a technique for data masking that not only creates a masked version of the data but also keeps the original data in a safe location. By doing this, we have created masked data tokens that cannot be connected back to the original data while maintaining access to it as needed. Simply said, we divided the statement into an array of words or tokens. A token may be a word, a punctuation mark, or a numeric value.

The command listed below can be used to accomplish this:

```
def tokenize(sentence):  
    return nltk.word_tokenize(sentence)
```

Stemming

Stemming is the process of removing all the word's suffixes, prefixes, and other word elements until only the lemma, or root, is left. As demonstrated in this project, we have derived various Natural language processing (NLP) advantages from stemming.

Using artificial intelligence, information extraction and retrieval and linguistic study in morphology include stemming (AI). Stemming and AI knowledge pull relevant information from massive sources like big data or the Internet since it may be necessary to look for different variations of a word associated with a topic to get the best results. Stemming is used by both queries and internet search engines.

To stem the dataset in simple steps, we:

1. Find the root word
2. For example: “organise”, “organising”, “organiser” all has the root word organ

```
def stem(word):  
    return stemmer.stem(word.lower())
```

In the next step, we create bags of words.

Bag of Words

The chaotic structure of text makes it challenging to model, and techniques like machine learning algorithms favour inputs and outputs with well-defined set lengths. Since the research used a text-based input dataset.

Machine learning algorithms can't use raw text directly; instead, the information needs to be converted into numbers. specifically, vectors in numbers. A bag-of-words model, or BoW for short, is a method for extracting features from text for use in modelling, such as with machine learning techniques. This is the reason we came up with a bag of words.

The technique is quite flexible and easy to use, and there are many different methods to use it to extract features from documents.

A bag-of-words is a textual illustration that shows where words appear in a manuscript.

There are two components:

- a collection of well-known words.
- a metric for the number of well-known words.

Because any information pertaining to the organization or structure of the words within the document is disregarded, it is referred to as a "bag" of words. The model is solely concerned with whether recognized phrases appear in the content, not with where in the document they do.

We create the bag of words by using the following command:

```
def bag_of_words(tokenized_sentence, words):  
    Sentence_words = [steam(word) for word in Tokenized_sentence]  
    Bag = np.zeros(len(word), dtype=np.float32)  
    for idx, w in enumerate(words):  
        If w in sentence_words:
```

```
Bag[idx] = 1
```

```
Return bag
```

Module 2 - Using NLTK and Numpy

Chatbots are one of the NLP applications that employ the Natural Language Toolkit, or NLTK. We will now use the NLTK library to enhance our rule-based chatbots.

Prior to running the pip command, please install the NLTK library.

pip install nltk

First thing is to import the library and classes we need to use.

```
import nltk
```

```
from nltk.chat.util import Chat, reflections
```

- Chat – Chat is a class that contains complete logic for processing the text data which the chatbot receives and find useful information out of it.
- Reflections – Another import we have done is reflections which is a dictionary containing basic input and corresponding outputs. We can also create your own dictionary with more responses you want. if you print reflections, it will be something like this.

Creating rules comes first after importing the libraries. A basic set of rules is created using the lines of code provided below. The user input, which was entered as a raw string, is described in the first line, followed by the chatbot's reply. These pairs can be changed to fit the questions and answers you wish.

For example,

```
pairs = [  
    [  
        r"my name is (.*)",  
        ["Hello %1, How are you today ?"],  
    ],  
    [  
        r"hi|hey|hello",  
        ["Hello", "Hey there"],  
    ],  
]
```

]

Following the development of rule pairs, we will define a function to launch the chat session. The function is quite straightforward, greeting the user and asking if they need any assistance. From here, the dialogue begins by calling a Chat class and sending it pairings and reflections.

```
def chat():  
    print("Hi! I am a chatbot created by Analytics Vidhya for your service")  
    chat = Chat(pairs, reflections)  
    chat.converse()  
    #initiate the conversation  
if __name__ == "__main__":  
    chat()
```

Using only Python and the NLTK library, we were able to build a fantastic Rule-based chatbot. The nltk.chat analyses various regex patterns found in user Intent and displays the output to the user accordingly. Let's launch the program and have a conversation with the chatbot you built.

4.2 ARCHITECTURE

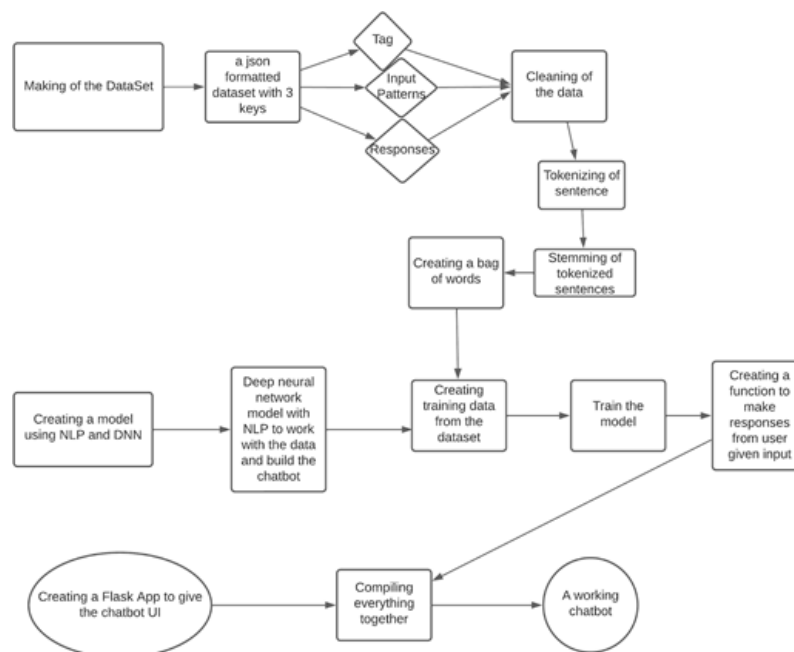


Fig 4.1: System Architecture

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED SYSTEM

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them - Anaconda is free and open source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create

visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

Creating a New Environment

We can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

We can replace <your_environment_name> with the name of our environment. After entering this command, conda will ask us if we want to proceed to which we should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if we want to create an environment with a particular version of Python, we should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, we can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, we can replace pack_name with the name of the package we want to use.

If we have a .yaml file, we can use the following command to create a new Conda

environment based on that file:

conda env create -n <your_environment_name> -f <file_name>.yml

We have also discussed how we can export an existing Conda environment to a .yml file later in this article.

Activating an Environment

We can activate a Conda environment by using the following command:

conda activate <environment_name>

We should activate the environment before we start working on the same. Also, replace the term <environment_name> with the environment name we want to activate. On the other hand, if we want to deactivate an environment use the following command:

conda deactivate

Installing Packages in an Environment

Now that we have an activated environment, we can install packages into it by using the following command:

conda install <pack_name>

Replace the term <pack_name> with the name of the package we want to install in our Conda environment while using this command.

Updating Packages in an Environment

If we want to update the packages present in a particular Conda environment, we should use the following command:

conda update

The above command will update all the packages present in the environment. However, if we want to update a package to a certain version, we will need to use the following command:

conda install <package_name>=<version>

Exporting an Environment Configuration

Suppose we want to share our project with someone else (colleague, friend, etc.). While we can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, we can create an environment configuration .yml file and share it with that person. Now, they can create an environment like our one by using the .yml file.

For exporting the environment to the .yml file, we'll first have to activate the same and run the following command:

```
conda env export ><file_name>.yml
```

The person we want to share the environment with only must use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If we want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if we want to uninstall a package from an activated environment, we'd have to use the following command:

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, we don't need to add a new environment but remove one. In such cases, we must know how to delete a Conda environment, which we can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

4.4 PROJECT MANAGEMENT PLAN

For this project implementation the first and foremost thing is to choose the best way to process the data using pre-built libraries or customizing the function as per

as requirements. Pandas, Nltk, Numpy, Sklearn are the prebuilt libraries which helps in achieving the desired result. All data possess different characteristics and all aspects of the data needed to be handled.

Integration of all these technologies gives user a smooth experience with data pre-processing and feature engineering. After all these processes the system must also chose the best feature out of all the features. Our system is based on statistics and machine learning. There will be some basic models on which the data will be tested and will give a rough idea about the performance of the dataset built.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

5.1.1 Business Problem Overview

In this case study we aim to automate some of these query resolutions with automation. This way when the patient asks a question, the chatbot itself that has been trained on such question-answer pairs can try to come up with an answer. If at all the patient is not satisfied with the answer generated by the chatbot, the chatbot itself will suggest some of the previously answered similar question-answer pairs. If the patient doesn't get his or her answer, then we may redirect him to an actual doctor.

This way we can ensure that both the doctors 'and patients' time is conserved. This is achieved by not overwhelming the doctor with a lot of patients' questions since they are filtered at each stage of the chatbot, the number of patients that still want to contact the doctor will reduce significantly. At the same time not making patients wait for the doctor to reply to the query significantly reduces the patient's waiting time and enables the patient to come up with a quick action in solving his/her problem.

5.1.2. ML Formulation of the Problem

There are two main components to solve the problem which might include many sub-components. Since at first, we are trying to generate an answer to the patient's query we can pose this problem as a seq2seq task trained in supervised fashion from previously answered question-answer pairs.

Secondly, when the patient is not satisfied with the answer provided, we must show the patient similar questions with answers that have been previously answered by a doctor, this we do by doing a semantic search task in an unsupervised fashion. Although we will train neural networks to generate question and answer embeddings which we use to do this semantic search. These

embeddings are generated using supervised learning binary classification approach which we will discuss as we progress through the blog.

5.1.3. Dataset Analysis:

For this task, we use the medical question answer dataset prepared by Lasse Regin Nelson. This dataset is uploaded in his GitHub <https://github.com/LasseRegin/medical-question-answer-data>. He has gathered such question answer pairs from prominent medical websites such as eHealth Forum, iCliniq, Question Doctors, WebMD where real doctors have provided public answers to the questions asked by patients. We thank the above websites and Lasse Regin for helping us with the dataset and enabling us to help the medical community further.

```
{
  "answer": "stopping smoking is about will power and being steadfast. you can stop safely by having bupropion or",
  "question": "how do i stop smoking now",
  "url": "http://ehealthforum.com/health/stop-smoking-question-t462882.html",
  "tags": [
    "addiction",
    "stop smoking"
  ]
},
```

FIG 5.1: Sample Data

The above picture shows a sample data provided in the GitHub. We were provided with about 25K question answer pairs. Along with the question answer pairs we were also given tags to efficiently categorize the questions into belonging to a particular disease, etc. We are also provided with the URLs from where we can find this question, answer pairs. The dataset is present in the json format which allows us to make our model trainable on a wide variety of similar datasets.

5.2 ALGORITHMS

5.2.1. Finetuning BioBert and Training Question Embedding

Extractor:

Data Preparation:

To understand how the BioBert model and the Question Embedding Extractor model is trained, let us first look at how the data for the same is prepared.

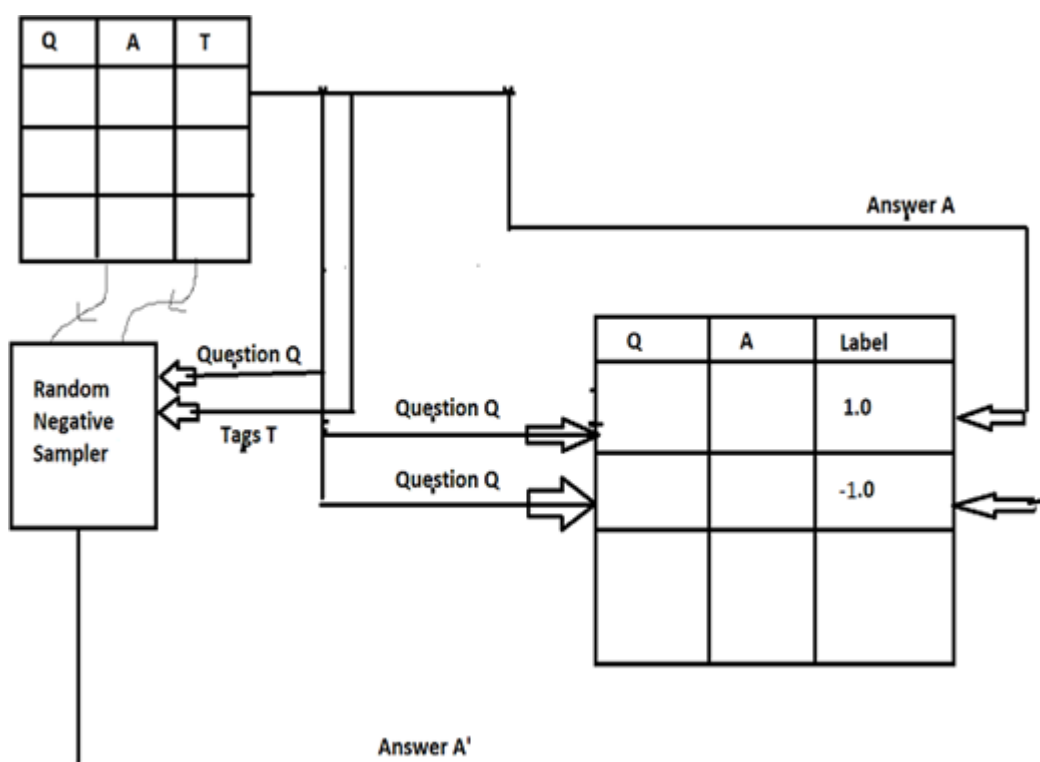


Fig 5.2: Data Preparation for Question Embedding Extractor Model

In our dataset, along with question and answers, we were also given tags which 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 take the question-answer pair as the positive label (1.0) and for generating negative samples, we use both the original Question Q and the corresponding Tags T and generate negative sample Q',A', T'. We generate this negative sample by randomly picking up a tuple and checking whether the intersection between T and T' is null. If the mentioned intersection is null, then we take this Answer A' and pair it with the original

Question Q and consider this as the negative sample. If the intersection is not null, then we must repeat the process of random sampling and checking for intersection. The below code implements the above.

Function to extract negative sample using data:

```
def extract_negative_samples(question,tags):
    stop=False
    while (not stop):
        sample_row= all_dataset.sample()
        sample_tags=sample_row.tags.values[0]
        inter_tags=set(tags[0]).intersection(set(sample_tags))
        if len(inter_tags)==0:
            stop=True
    return sample_row

#preparing the negative labelled dataset
from tqdm.notebook import tqdm
tqdm.pandas()
negative_labels=all_dataset.progress_apply(lambda x:
pd.Series([x.short_question,extract_negative_samples(x.short_ques
tion,x.tags).short_answer.values[0],x.tags]),axis=1)
negative_labels['label']=-1.0
negative_labels.columns=['short_question','short_answer','tags','label']
```

Data Preprocessing:

There are a few other minor details that we need to discuss about how we have pre-processed the question-and-answer text data. Pre-processing of the given question and answer data consisted of de-contracting some of the words such as “won’t” to “will not”, remove unnecessary symbols such as! etc. We also do not want to pad most of the question and answer with the padding constant. Therefore, when we calculated different percentiles of both question-and-answer data, we found that about 100% of all

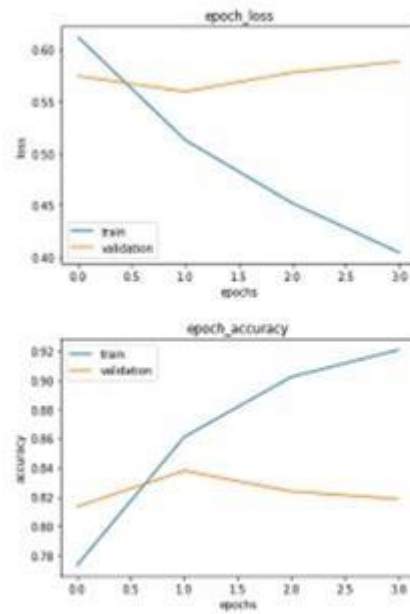
questions data and 99% of all answers data lie within 500 words. So, post tokenizing, we truncated our sequences to a max sequence length of 512, this way we can use both the default pre-trained weights of Bio-Bert and we do not lose any information. We also needed to pass attention mask to the Bio-Bert model so that it understands that it must focus on the actual content rather than on the padding token.

To implement this, we have written code for preparing BioBert Train Data. Now, we have our data ready let us see how we have defined the question extractor model, answer extractor model and how we fine-tuned the pre-trained hugging-face BioBert transformer to achieve the task of classifying whether a given question and answer relate to the same context.

Model Creation and Training:

First we will write the code for medical QA Model with Bert and then by that code we will analyze. We can observe how we have created the MedicalQAModelwithBert which consists of a shared pre-trained BioBert Model and two residual blocks with dropout each for question and answers data. We used residual blocks so that we want to enable the model to directly pass the BioBert embeddings in case it finds to be best. The dropout makes sure that we can overcome overfitting as far as possible. After we get the embeddings for both question and answer after passing through the respective residual blocks, we then calculate the cosine similarity between them.

The model is then trained using mean squared error loss between the expected cosine similarities -1.0 and 1.0 and the cosine similarities returned by the model. We chose the best model that performed well on both train and validation data, after trying with various thresholds to classify between positive and negative classes.



BioBERT Training epoch vs loss and accuracy

Fig 5.3: BioBERT epoch vs loss and accuracy

Model Evaluation on Validation Data:

We compared the cosine similarities returned by the model for the validation dataset for both correctly classified negative and positive points, as well as the cosine similarities returned by the model for misclassified points that are originally negative and positive points. Please have a look at the below plots for the same.

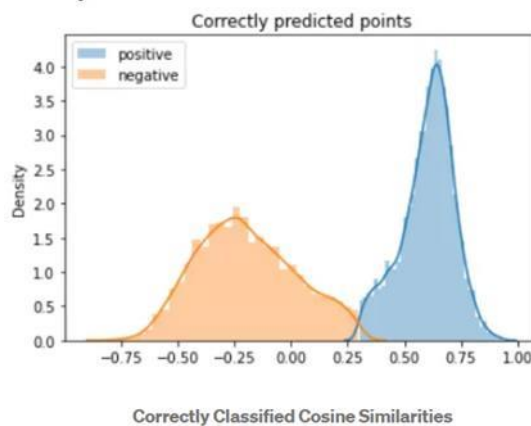


Fig 5.4: Comparison of cosine similarities of correctly classified negative and positive points

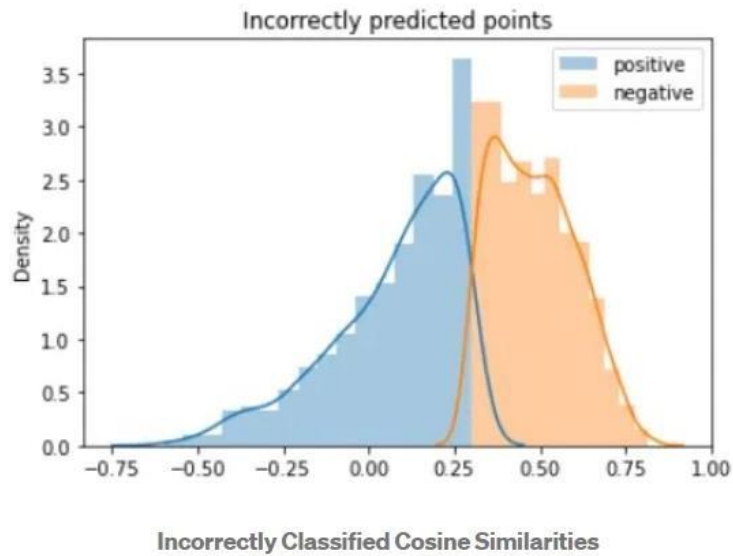
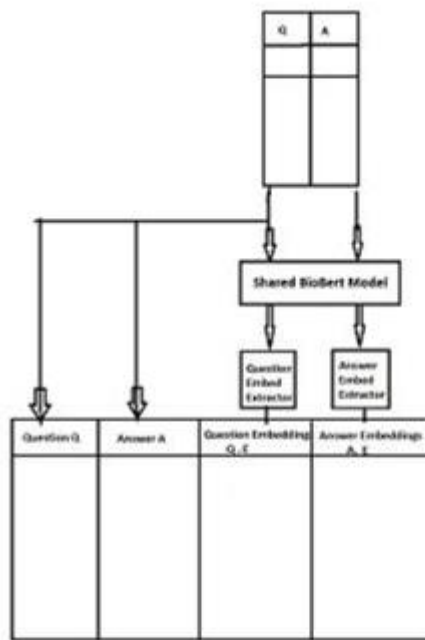


Fig 5.5: Comparison of cosine similarities of incorrectly classified negative and positive points

From the above plots, we can observe that the correctly classified negative and positive points are well separated from each other while the incorrectly classified positive and negative points are very close to the threshold of 0.3 for which the model gave best train and validation accuracy. These two plots indicate that our model when differentiating correctly makes sure that we are well able to cosinely separate positive and negative points and when it fails to classify, it only does by a small amount.

Extracting Question and Answer Embeddings from the Model:

Now given that we trained the model, let us look at how we extract the question and answer embeddings from the mode. We pass the question and answers to the fine-tuned BioBert model and pass it to both the residual blocks of question and answer and then we store both these embeddings along with the question and answers to be used for semantic search.



Extracting Question and Answer Embeddings

Fig 5.6 Extracting Question and Answer Embeddings

5.2.2. Fine-Tuning the GPT2 Model:

Data Preparation:

Having trained the BioBert Model and extracting the question and answer embeddings, let us look at how we are going to prepare the data for training our GPT2 model.

Our first task is, for each patient question in the train dataset we do a semantic search to fetch similar answers and the corresponding questions. To do this task we use FAISS library from Facebook. We are doing cosine similarity FAISS search using the pre-computed embeddings of each answer in the dataset and the embeddings of the current question. FAISS along with giving similar results, gives a score on to how similar they are and gives them in sorted fashion.

Now once we get the ranked similar question-answer pairs to the given question and answer in the train dataset, we combine these ranked similar question-answer pairs Q1,A1,Q2,A2,.. with the original question Q and

corresponding answer A to get the following string.

“..`Question: Q2 `Answer: A2 `Question: Q1 `Answer: A1 `Question: Q
`Answer: A”

Now they are two things that we need to which we need to pay attention. First, notice how we explicitly mentioned the start of question and answer with special strings “`Question” and “`Answer”. These allow the language model to learn that once it encounters “`Question” it should expect a question and the same applies for answer as well.

The next thing we need to observe is that the most similar question-answer pair Q1-A1 is more closer to the original question-answer pair Q-A than Q2-A2. This allows our model to have a better recent context while learning the language model. Since our pre-trained GPT-2 Model was trained using max sequence length of 1024. We only take the last 1024 tokens, post tokenizing. In this process, we can observe that since question-answer pair with greater similarity lies close to the original question, whatever context is lost will be from the relatively less similar question-answer pairs. Hence our format helps solving this data loss issue as well. The below figure shows the data preparation pipeline for GPT-2.

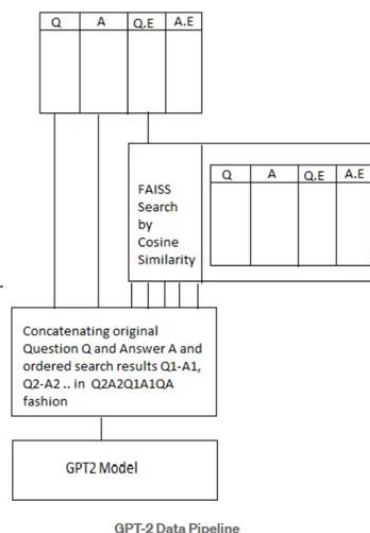


Fig 5.7: GPT-2 Data Pipeline

Since we know that GPT-2 learns the language model by learning to predict the next word by taking the entire sentence before the word as context, we

have to make sure that GPT-2 model is aware that when we give Q2A2Q1A1QA to the model it should learn to predict A taking Q2A2Q1A1Q as context. To do this we calculate 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 taking both the question and the previous similar question-answer pairs data.

GPT-2 Training:

Now we have our dataset ready, let us see how we are going to train our GPT-2 model with the custom loss function.

By writing the code of loading pre trained gpt2 tokenizer and gpt2 model.

We can see how simple it is to train the hugging face GPT-2 model using TF gradient. Even though the batch size of dataset is 1, updating the gradients every 50 iterations made it easy for us to train with higher batch size in spite of having memory limitations.

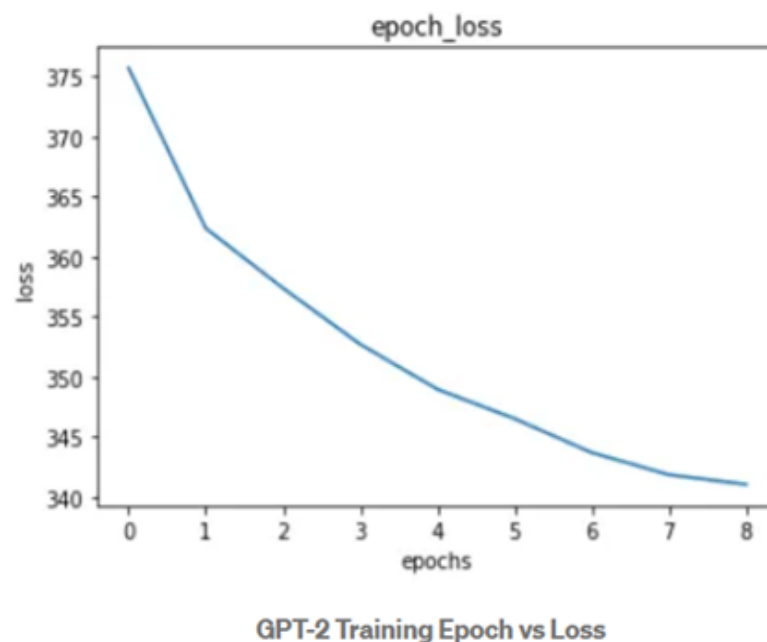


Fig 5.8: GPT-2 epoch vs loss

GPT-2 Model Evaluation:

We generated answers for 5 questions each from both train and validation

dataset using the model that gave least train loss. Although the bleu score is high for some of the answers and low for some other, overall we were able to generate decent looking answers for most of the questions. Please have a look at both the training and validation answers that have been generated. Note: The first line is the entire context, the second line is the actual answer and the third line is the generated answer in each paragraph.

```

can tell you it took a long time for the injury to heal because we use our hands for everything but with time i am able to still use my hand and
: i think its gonna be better if you add another finger with a plastic surgery instead of the amputated one but to get the other fingers closer it
: hi this is a very common problem that can be due to a joint injury or a joint injury that may be due to a joint injur
-----
ered myself and i felt something unusual i went all the way in with my entire index finger around 3 inches deep i felt a hard kind of pointy bump
: hi just a myth the only way cancer spreads quickly is if it is diagnosed late or that is very vigorous do hope this helps good luck
: hi if you have prostate cancer then you should be fine if you have a prostate cancer then you should be fine if you hav
-----
iacin and fish oil my hdl is now and has been for several years 40 47 my ldl and triglycerides are below 100 and are around 80 90 my internal doct
: congratulations it sounds like you are doing an excellent job with your lifestyle changes diet and exercise are the cornerstones of treating bot
: i am sorry to hear about your situation but i am sorry to hear about your situation but i am not sure what you are goi
-----
to be continued `ANSWER: ear ringing and fatigue i started amoxicillin the next day and the sinus pressure went away but the rest of my sympto
: most probably it is an ear canal infection you need to get antibiotic course also to get examined by ent do you have spinning or dizziness disci
: hi this is due to a bacterial infection that is causing ear pain and inflammation in the ear it is not a bacterial infec
-----
her eyes a few moments later hi there thank you for answering my question no when i instill them she is lying on her side and she feels them go t
: it can be true at times if you have strong healthy teeth you should not have a problem from chewing ice but teeth that are weakened by old filli
: hi this sounds like a joke but i do not think it is true it is true that you can not chew gum or gum gum is not a
-----

```

Fig 5.9: Training Samples

```

to healthcare gov and begin the process `QUESTION: i have medicare now my husband has none how does that work with the new health insurance
: you can shop for coverage before january 1 2014 but insurance will not take effect before then
: you can not get insurance under the ac a however you can still get insurance under the medic
-----
ples with fulid `QUESTION: constant masturbation cause bumps on penis i damaged the skin area where the bumps seem to have formed so i noticed a
: this could be a condition called hidradenitis suppurativa which could occur at any time it is a scarring condition of recurrent boils in the gr
: i am sorry to hear of your child is health problems but i do not know why you are having boils or boils but i do not
-----
warts so he is safe to pet them or get licked as much as he wants my question my little dog has a wart like breakout to of head back leg with ar
: that would be highly unlikely since those tools a vaginal speculum would need to be unwashed unsterilized and still wet from a previous patient
: hi this is a very common form of herpes simple x herpes simple x herpes simple x herpes simple x herpes simple x herpes sin
-----
the zaps and dizziness now til after i take the celexa in the morning time insomnia and light headed just dont know if the celexa is causing a a
: it is not safe to take 5400 mg of ibuprofen per day so you would be wise to see a rheumatologist for a better and safer way to control your ost
: hi this is a very common problem with cort is one and it can be due to a combination of medications and medications that
-----
available equivalent montelukast i have noticed an increase of sweating and tiredness since switching over to the generic i am going to stop taki
: during the meals this medication should be taken with food in your stomach
: i do not know of any over the counter drugs that would be
-----

```

Fig 5.10: Validation Samples

Put It all Together:

Having seen the final inference pipeline and how each individual component is trained, let us look through code how when given a patient question it passes to the question extractor model to get the question embeddings, how these embedding are used to fetch similar question-answer pairs, post

obtaining these question-answer pairs how they can be used to generate the answer to the question.

5.3 TESTING

There are different components in this task which require different performance metrics. For the task of seq2seq generation, instead of relying on bleu score for measuring the efficiency of our model, we rather evaluate it by the manual interpretation of the quality of answers generated. The reason for not using bleu score is, even though it is generally used for evaluating seq2seq tasks, it is still not highly reliable for us to judge our model on this bleu score, at least for our biomedical task, as for some sentences it may give a very good bleu score and for some others it might give very low bleu scores. To better judge the model, the answers generated by the model should be open sourced to be judged by medical experts, only then we can come up with a correct evaluation of the model.

For the task of semantic search although we do not have a labelled dataset indicating which questions are similar to calculate the overall metric, we use cosine similarity to evaluate the embeddings of question and answer returned by the model.

CHAPTER 6

RESULTS AND DISCUSSION

- Medical chatbots that are automated and designed for conversations have ability to cut down healthcare expenses while providing better access to medical services and information through technology.
- We developed a bot that interacts with patients regarding their medical issues and provides personalized diagnosis based on the symptoms they present and other relevant factor.
- We created a diagnostic chatbot that interacts with patients and addresses their medical concerns by offering personalized diagnosis based on their symptoms and medical history.
- After analyzing the symptoms that were gathered, accurate ones were determined with a 86% rate of successfully identifying them and a precision level of 95%.
- Instead of just returning the generated answer, we have also displayed the ranked previously answered question-answer pairs that are similar to the given user question.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

The suggested system's goal is to make the healthcare chatbot more approachable to its intended audience. Only users who have mastered the keyboard shortcuts for their native language can use the software. Disease and symptom predictions are all that a health chatbot can offer. By adding support for more medical aspects and depth of symptoms, and by providing more specifics regarding symptom description, future generations of the chatbot can be made more efficient at symptom recognition, and analysis performance is likely to be significantly taken forward. Due in large part to the training data, AI algorithms play a crucial role in the implementation of personalized health professionals. Finally, widespread scientific education and the use of personalized medicine could save many lives. Because consumers are spending more time in messaging applications than any other apps, they are the technology of the future. As a result, the potential for the medical chatbot to make a difference is both now and in the future. This medical communication may be available to humanity anywhere in the universe. All they need is access to the internet, which can be provided by any basic computer or mobile phone. For the medical chatbot to be able to handle all types of disorders, its efficiency is generally improved by adding greater combination of phrases and expanding the usage of database. Voice communication can also be provided within the system to make it more user-friendly.

7.2 FUTURE WORK

In the future, we can experiment on how a transformer model can work on solving this problem directly as a seq2seq problem. Even if transformer model can directly generate answers to the given question, we still need BioBert to extract similar patient question-answer pairs. We can also work on gathering more biomedical data so that the model doesn't overfit to the given data and can generate more appropriate answers.

7.3 RESEARCH ISSUES

Medical chatbots are designed to provide users with quick and convenient access to health-related information, advice, and guidance.

However, there are several research issues associated with the development and use of medical chatbots, including:

1. **Accuracy and reliability:** Medical chatbots must be accurate and reliable in order to provide users with trustworthy information. This requires careful programming and testing to ensure that the chatbot is able to understand and respond appropriately to user queries.
2. **Privacy and security:** Medical chatbots often deal with sensitive information about users' health, which raises important privacy and security concerns. Developers must ensure that the chatbot is compliant with relevant data protection regulations and that user data is kept secure.
3. **Ethics and accountability:** Medical chatbots are programmed to provide advice and guidance, but they may not always be able to fully understand the context or complexity of a user's situation. Developers must consider the ethical implications of providing advice that could potentially harm users, and ensure that the chatbot is held accountable for its actions.
4. **User experience and accessibility:** Medical chatbots must be easy to use and accessible to a wide range of users, including those with disabilities or who speak different languages. Developers must consider issues such as user interface design, natural language processing, and multi-language support.
5. **Integration with healthcare systems:** Medical chatbots may need to be integrated with existing healthcare systems, such as electronic health records or telemedicine platforms. Developers must consider issues such as interoperability and data sharing to ensure that the chatbot can be used effectively within the healthcare ecosystem.
6. **Training and testing:** Medical chatbots require ongoing training and testing to ensure that they are up-to-date and accurate. Developers must consider issues such as data sources, machine learning algorithms, and testing methodologies to ensure that the chatbot is able to provide accurate and reliable information over time.

7.4 IMPLEMENTATION ISSUES

Medical chatbots can be a useful tool for providing healthcare services to patients, but there are several implementation issues that need to be considered. Here are some key implementation issues for medical chatbots:

Data privacy and security: Medical chatbots may collect and store sensitive personal health information. It is essential to ensure that chatbots comply with relevant data protection regulations, such as HIPAA in the US or GDPR in the EU.

Accuracy and reliability: Medical chatbots need to be accurate and reliable to ensure that they provide correct medical advice and diagnoses. They should be designed to detect and correct errors, and any limitations should be clearly communicated to users.

Training and testing: Medical chatbots should be thoroughly tested and validated before deployment to ensure that they function as intended. This includes testing for accuracy, reliability, and user satisfaction. Additionally, chatbots should be continuously monitored and updated to ensure they remain accurate and relevant.

Integration with existing systems: Medical chatbots should be integrated with existing healthcare systems, such as electronic health records and telehealth platforms, to ensure a seamless patient experience. This requires careful planning and coordination between the chatbot development team and healthcare providers.

User experience: Medical chatbots should be designed to provide a user-friendly experience. This includes designing intuitive user interfaces, providing clear and concise instructions, and using natural language processing to facilitate communication between users and the chatbot.

Ethical considerations: Medical chatbots raise ethical considerations, such as

issues around patient autonomy, informed consent, and potential biases in the chatbot's decision-making algorithms. These issues should be addressed and mitigated to ensure that chatbots are used ethically and responsibly.

Overall, the implementation of medical chatbots requires careful planning, testing, and ongoing monitoring to ensure that they provide safe, accurate, and effective healthcare services.

REFERENCES

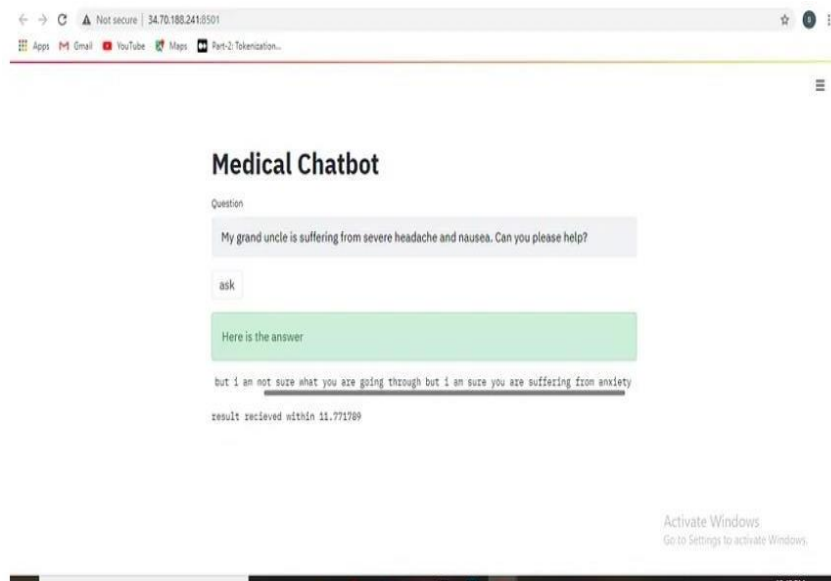
- [1] 8th International Conference on Reliability, Infocom Technologies, and Optimization (Trends and Future Directions), 2020, L. Athota, V. K. Shukla, N. Pandey, and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence."
- [2] 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 2018, pp. 1-5, doi: 10.1109/TIMES-iCON.2018.8621678, N. Rosruen and T. Samanchuen, "Chatbot Utilization for Medical Consultant System."
- [3] "Artificial Intelligence based Chat Bot for Patient Health Care," 2022 International Conference on Computer Communication and Informatics (ICCCI), P. Anandan, S. Kokila, S. Elango, P. Gopinath, and P. Sudarsan.
- [4] According to M. G., "Mega Bot - The Healthcare Chatbot," presented at the Second International Conference on Electronics and Sustainable Communication Systems (ICESC) in 2021.
- [5] A. Soufyane, B.A. Abdelhakim, and M.B. Ahmed (2021). An Advanced Medical Chatbot Employing Natural Language Processing and the TF-IDF Algorithm for Text Understanding. Ben Ahmed, M., S. Mellouli, L. Braganca, B. Anouar Abdelhakim, and K.A. Bernadetta (eds.), Emerging Trends in ICT for Sustainable Development. Developments in Science, Engineering, and Innovation.
- [6] Second International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022, "Machine Learning Based Healthcare Chatbot," R. Goel, R. P. Goswami, S. Totlani, P. Arora, R. Bansal, and D. Vij.
- [7] Those three authors are Ayanouz, S.; Abdelhakim, B. A.; and Benhmed, M. (2020). For use in the medical field, here is a chatbot design that uses natural

language processing and machine learning to provide the best possible service. For inclusion in the Third International Conference Proceedings on Networking, Information Systems, and Security. To be held in Tokyo, Japan, NISS2020 is the third annual international conference on networking, information systems, and security. <https://doi.org/10.1145/338672.3387897> ACM.

[8] To cite this article: Regin, D. R., Rajest, D. S. S., T. S., G. J. A. C., & R. S. (2022). Python-Based Natural Language Processing (NLP) Chatbot for Automated Conversation. Three and a half years later, 314-336 appeared in the third issue of the Central Asian Journal of Medical and Natural Science.

APPENDIX

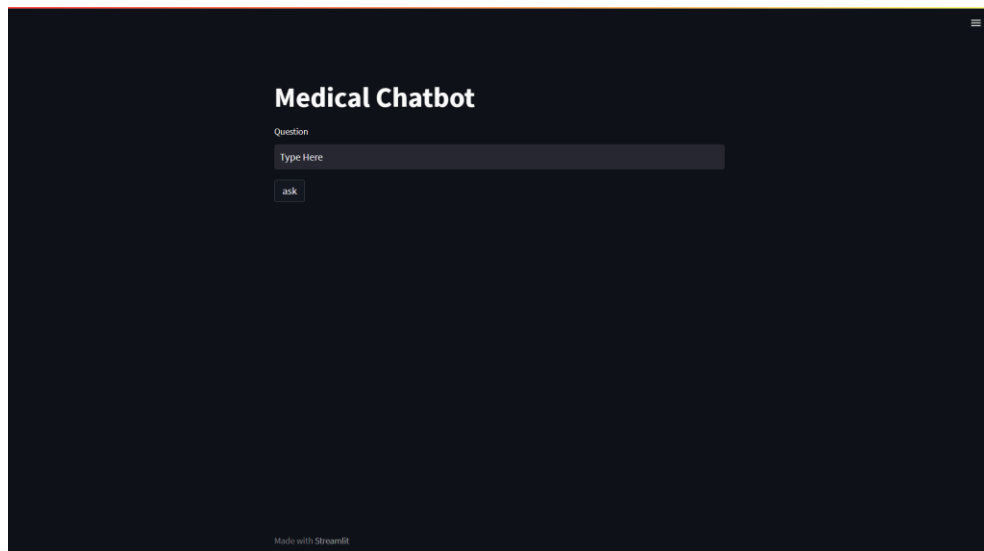
A. SCREENSHOTS



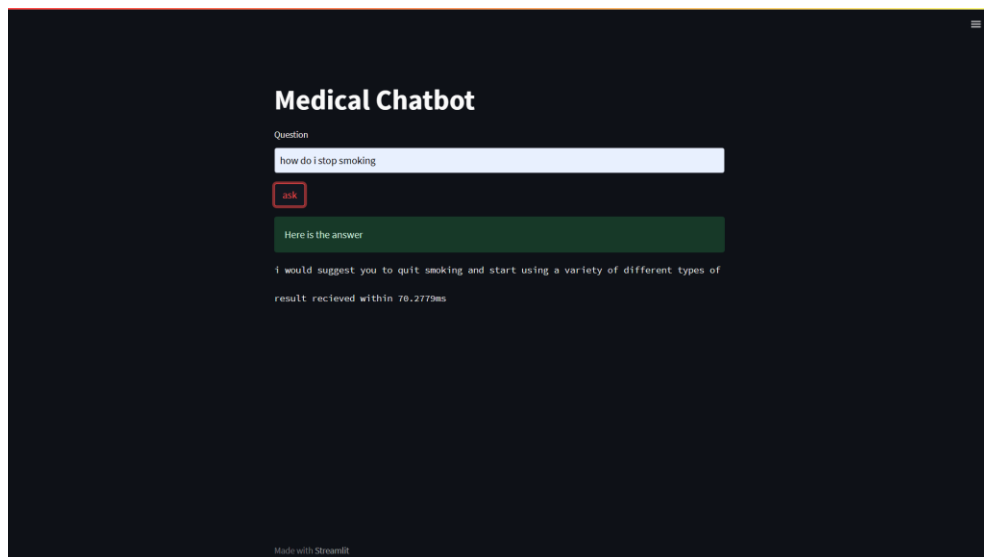
Instead of just returning the generated answer, we have also displayed the ranked previously answered question-answer pairs that are similar to the given user question. Please have a look at the below image for the same.



Alternate Sample Input and Output



Front end made and customized using streamlit library in python. We have displayed the previously ranked answered pairs that are similar to the given user question.



Source Code:

Part 1: BioBERT Training after data preprocessing

```
In [4]: #creating the medicalbert model
#https://github.com/ash3n/DocProduct/blob/master/docproduct/models.py
class MedicalQAModelWithBert(tf.keras.Model):
    def __init__(
        self,
        trainable=False,
        name=''):
        super(MedicalQAModelWithBert, self).__init__(name=name)

        self.q_ffn_layer = FFN(name='q_ffn')
        self.a_ffn_layer = FFN(name='a_ffn')
        self.biobert_model=biobert_model
        self.biobert_model.trainable=trainable
        self.cos=tf.keras.layers.Dot(axes=1,normalize=True)

    def call(self, inputs):
        question_embeddings=self.biobert_model(input_ids=inputs['question'],attention_mask=inputs['question_mask']).pooler_output
        answer_embeddings=self.biobert_model(input_ids=inputs['answer'],attention_mask=inputs['answer_mask']).pooler_output
        q_ffnn=self.q_ffn_layer(question_embeddings)
        a_ffnn=self.a_ffn_layer(answer_embeddings)
        output=self.cos([q_ffnn,a_ffnn])
        return {"label":output}
```

```
In [ ]: #compiling and training the medical qa model
import tensorflow.keras.backend as K
K.set_floatx('float32')
learning_rate=5e-6
num_epochs=5
optimizer = tf.keras.optimizers.Adam(lr=learning_rate)
medical_qa_model=MedicalQAModelWithBert(trainable=True)
medical_qa_model.compile(
    optimizer=optimizer, loss=tf.keras.losses.mean_squared_error, metrics=[custom_metric_acc])

epochs = num_epochs

medical_qa_model.fit(dataset,validation_data=val_dataset,epochs=epochs,callbacks=[custom_callback()])
medical_qa_model.summary()
```

The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.
Epoch 1/5

The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.
The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.
The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.

23801/23801 [=====] - ETA: 0s - loss: 0.6119 - custom_metric_acc: 0.7737

The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.
The parameters 'output_attentions', 'output_hidden_states' and 'use_cache' cannot be updated when calling a model.They have to be set to True/False in the config object (i.e.: 'config=XConfig.from_pretrained('name', output_attentions=True)').
The parameter 'return_dict' cannot be set in graph mode and will always be set to 'True'.

23801/23801 [=====] - 7862s 327ms/step - loss: 0.6119 - custom_metric_acc: 0.7737 - val_loss: 0.5745 - val_custom_metric_acc: 0.8136

Epoch 2/5
23801/23801 [=====] - 7781s 327ms/step - loss: 0.5128 - custom_metric_acc: 0.8611 - val_loss: 0.5595 - val_custom_metric_acc: 0.8380

Epoch 3/5
23801/23801 [=====] - 7784s 327ms/step - loss: 0.4518 - custom_metric_acc: 0.9020 - val_loss: 0.5779 - val_custom_metric_acc: 0.8239

Epoch 4/5
23801/23801 [=====] - 7785s 327ms/step - loss: 0.4042 - custom_metric_acc: 0.9204 - val_loss: 0.5885 - val_custom_metric_acc: 0.8187

Epoch 5/5
3040/23801 [==>.....] - ETA: 1:44:55 - loss: 0.3494 - custom_metric_acc: 0.9488

```
#finding the best accuracy and the corresponding threshold of the model of best validation accuracy
best_threshold_acc(predicted_labels)
```

```
HBox(children=(FloatProgress(value=0.0, max=21.0), HTML(value='')))
accuracy for threshold -1.0 is 0.4999579867238047
accuracy for threshold -0.9 is 0.4999579867238047
accuracy for threshold -0.8 is 0.4999579867238047
accuracy for threshold -0.7000000000000001 is 0.5008822788001008
accuracy for threshold -0.6000000000000001 is 0.5060919250483152
accuracy for threshold -0.5 is 0.5318040500798252
accuracy for threshold -0.4 is 0.5820519284093774
accuracy for threshold -0.30000000000000004 is 0.6506175951600706
accuracy for threshold -0.2 is 0.7279220233593816
accuracy for threshold -0.1 is 0.7923703890429376
accuracy for threshold 0.0 is 0.8386690194101336
accuracy for threshold 0.1 is 0.8675741534324847
accuracy for threshold 0.2 is 0.8825308797580035
accuracy for threshold 0.30000000000000004 is 0.8852197294345013
accuracy for threshold 0.4 is 0.8705150827661541
accuracy for threshold 0.5 is 0.8350558776573397
accuracy for threshold 0.6000000000000001 is 0.7401899000084027
accuracy for threshold 0.7000000000000001 is 0.5780186538946307
accuracy for threshold 0.8 is 0.5128140492395598
accuracy for threshold 0.9 is 0.5008822788001008
accuracy for threshold 1.0 is 0.5000420132761952
```

```
best accuracy is 0.8852197294345013 at threshold 0.30000000000000004
```

```
train_best_threshold_acc(train_predicted_labels)
```

```
HBox(children=(FloatProgress(value=0.0, max=21.0), HTML(value='')))
accuracy for threshold -1.0 is 0.5000105035396929
accuracy for threshold -0.9 is 0.5000105035396929
accuracy for threshold -0.8 is 0.5000735247778502
accuracy for threshold -0.7000000000000001 is 0.5016490557317816
accuracy for threshold -0.6000000000000001 is 0.5125307228536017
accuracy for threshold -0.5 is 0.5525912232422326
accuracy for threshold -0.4 is 0.6247085267735227
accuracy for threshold -0.30000000000000004 is 0.7173497468646934
accuracy for threshold -0.2 is 0.807281053715102
accuracy for threshold -0.1 is 0.8768354935613302
accuracy for threshold 0.0 is 0.9200260487784383
accuracy for threshold 0.1 is 0.9439741192781967
accuracy for threshold 0.2 is 0.9555910341785182
accuracy for threshold 0.30000000000000004 is 0.9585950465306808
accuracy for threshold 0.4 is 0.9496460307123501
accuracy for threshold 0.5 is 0.917505199252148
accuracy for threshold 0.6000000000000001 is 0.8139823120391572
accuracy for threshold 0.7000000000000001 is 0.6129235552381153
accuracy for threshold 0.8 is 0.5143793458395479
accuracy for threshold 0.9 is 0.5006827300800369
accuracy for threshold 1.0 is 0.4999894964603071
```

```
best accuracy is 0.9585950465306808 at threshold 0.30000000000000004
```

Question Extractor Model extraction:

```
1 #defining the question extractor model
x1=tf.keras.layers.Input((512),name='question',dtype='int64')
x2=tf.keras.layers.Input((512),name='question_mask',dtype='int64')
q_embed=medical_qa_model.get_layer('q_ffn')(medical_qa_model.get_layer('tf_bert_model').bert(input_ids=x1,attention_mask=x2).pooler_output)
question_extractor_model=tf.keras.Model(inputs=[x1,x2],outputs=q_embed)
```

```
2 #saving the question extractor model to disk
question_extractor_model.save('question_extractor_model_2_11')
```

```
3 #Loading the question extractor model from disk
question_extractor_model=tf.keras.models.load_model('question_extractor_model_2_11')
```

WARNING:tensorflow:No training configuration found in save file, so the model was *not* compiled. Compile it manually.

```
4 #function to extract embeddings given a question and a question mask
def question_extractor(sam_dict):
    q_embed=question_extractor_model({ 'question':sam_dict['question'], 'question_mask':sam_dict['question_mask']})
    return q_embed
```

```
5 #defining function to extract question embeddings given question
def extract_question_embed(question):
    max_length=512
    tokenized_questions=[]
    tokenized_question = biobert_tokenizer.encode(question)
    tokenized_questions.append(tokenized_question)

    # padding the sequences
    tokenized_questions = tf.keras.preprocessing.sequence.pad_sequences(
        tokenized_questions, maxlen=max_length, padding='post')
    attention_mask=[[1 if token>0 else 0 for token in q] for q in tokenized_questions]
    return question_extractor({ 'question':np.array(tokenized_questions), 'question_mask':np.array(attention_mask)})
```

```
6 #defining the answer extractor model
x1=tf.keras.layers.Input((512),name='answer',dtype='int64')
x2=tf.keras.layers.Input((512),name='answer_mask',dtype='int64')
a_embed=medical_qa_model.get_layer('a_ffn')(medical_qa_model.get_layer('tf_bert_model').bert(input_ids=x1,attention_mask=x2).pooler_output)
answer_extractor_model=tf.keras.Model(inputs=[x1,x2],outputs=a_embed)
```

```
2] #saving the answer extractor model to disk
answer_extractor_model.save('answer_extractor_model_2_13')
```

WARNING:absl:Found untraced functions such as embeddings_layer_call_and_return_conditional_losses, embeddings_layer_call_fn, encoder_layer_call_and_return_conditional_losses, encoder_layer_call_fn, pooler_layer_call_and_return_conditional_losses while saving (showing 5 of 1075). These functions will not be directly callable after loading.

WARNING:absl:Found untraced functions such as embeddings_layer_call_and_return_conditional_losses, embeddings_layer_call_fn, encoder_layer_call_and_return_conditional_losses, encoder_layer_call_fn, pooler_layer_call_and_return_conditional_losses while saving (showing 5 of 1075). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: answer_extractor_model_2_13/assets

INFO:tensorflow:Assets written to: answer_extractor_model_2_13/assets

```
3] #Loading the answer extractor model from disk
answer_extractor_model=tf.keras.models.load_model('answer_extractor_model_2_13')
```

WARNING:tensorflow:No training configuration found in save file, so the model was *not* compiled. Compile it manually.

WARNING:tensorflow:No training configuration found in save file, so the model was *not* compiled. Compile it manually.

```
4] #function to extract embeddings given an answer and an answer mask
def answer_extractor(sam_dict):
    a_embed=answer_extractor_model({ 'answer':sam_dict['answer'], 'answer_mask':sam_dict['answer_mask']})
    return a_embed
```

```
7] #defining function to extract answer embeddings
def extract_answer_embed(answer):
    max_length=512
    tokenized_answers=[]
    tokenized_answer = biobert_tokenizer.encode(answer)
    tokenized_answers.append(tokenized_answer)

    # padding the sequences
    tokenized_answers = tf.keras.preprocessing.sequence.pad_sequences(
        tokenized_answers, maxlen=max_length, padding='post')
    attention_mask=[[1 if token>0 else 0 for token in q] for q in tokenized_answers]
    return answer_extractor({ 'answer':np.array(tokenized_answers), 'answer_mask':np.array(attention_mask)})
```

Part 2: Training GPT-2 Model after initializing the environment and above extracted model

```
In [ ]: #training the gpt2 model
learning_rate=0.0001
optim=tf.optimizers.Adam(learning_rate)
j=0
loss_cum=0
epoch_loss=0
stop_training=False
epochs=10
for epoch in range(epochs):
    print('epoch',epoch)
    for i in tqdm(gpt_train_dataset):
        with tf.GradientTape() as tape:
            context=i[0]['context']
            loss_mask=tf.cast(i[0]['loss_mask'],tf.float32)
            output=tf_gpt2_model(context,training=True)['logits']
            loss_value = tf.reduce_sum(tf.nn.sparse_softmax_cross_entropy_with_logits(
                labels=np.array(i[0]['label']), logits=output[:, :-1])*loss_mask)

            j=j+1
            loss_cum+=loss_value
            epoch_loss+=loss_value/100
            if j%50==0:
                print(loss_cum/50)
                grads=tape.gradient(loss_cum/50,tf_gpt2_model.trainable_variables)
                optim.apply_gradients(zip(grads,tf_gpt2_model.trainable_variables))
                loss_cum=0
            if tf.math.is_nan(loss_value):
                print('loss became nan')
                stop_training=True
                break
            if j%10000==0:
                tf_gpt2_model.save_pretrained("tf_gpt2_model_2_10"+str(epoch)+'_'+str(j))
    print('epoch_loss by 100',epoch_loss/237)
    epoch_loss=0
    if stop_training:
        break
    tf_gpt2_model.save_pretrained("tf_gpt2_model_2_10"+str(epoch))

epoch 0
HBox(children=(FloatProgress(value=0.0, max=23783.0), HTML(value='')))
tf.Tensor(364.4527, shape=(), dtype=float32)
tf.Tensor(370.41684, shape=(), dtype=float32)
tf.Tensor(351.39758, shape=(), dtype=float32)
tf.Tensor(388.43634, shape=(), dtype=float32)
```

```

1: #function to return sample original and generated answers for the train dataset
def return_sample_output_predicted():
    sample_data=next(iter(gpt_train_dataset))
    v=sample_data[0]['context'].numpy()[0]
    mask_start = len(v) - list(v[::-1]).index(4600) + 1
    input=v[mask_start+1:]
    next_step_input=[]
    input1=list(input)+next_step_input
    final_output=[]
    for i in range(0,1024-len(input)):
        input1=list(input1)+next_step_input
        output=tf_gpt2_model([np.array(input1)]['logits'])
        output1=np.argmax(output,axis=1)
        next_step_input=[output1[-1]]#appending last output to input
        next_step_char=tokenizer.decode(next_step_input)
        final_output.append(next_step_char)
    expected_output=v[mask_start+1:]
    return v,tokenizer.decode(expected_output),' '.join(final_output)

1: #displaying sample original context, expected answer and generated answer for train dataset
for i in range(5):
    v,exp,fin=return_sample_output_predicted()
    print(tokenizer.decode(v))
    print(exp)
    print(fin)
    print('-----')

```

can tell you it took a long time for the injury to heal because we use our hands for everything but with time i am able to still use my hand and the pain is gone splint the finger and keep it on for a while `QUESTION: what causes pain when you lower your arm after raising it i injured my arm lifting a box of books had an xray which showed only some arthritis have had some problem with pain in my shoulder when sleeping on it in the past it is my right shoulder `ANSWER: i have had surgery on my rotator cuff because of natural wear i had a problem that if i raised and then lowered my arm the shoulder hurt when i lowered my arm it turned out to be shoulder rotator cartilage problem the pain disappeared after a few moments after i lowered my arm `QUESTION: what are the different surgical options for treatment of arthritis of the hand and wrist `ANSWER: treatmentsurgery for painful arthritic fingers and how to prevent further deterioration `QUESTION: arthritis on its own or arthritis as a result of an inflammatory bowel disease i was diagnosed with arthritis when i was twelve i am now nineteen and am seeing a doctor this summer because i have developed symptoms of an inflammatory bowel disease i know that arthritis can develop as a symptom of inflammatory bowel diseases how do i know if i have arthritis and an inflammatory bowel disease or if i just have arthritis as a symptom of an inflammatory bowel disease is there a distinction between the two in treatments `ANSWER: arthritis on own `QUESTION: what are the different surgical options for treatment of arthritis of the hand and wrist `ANSWER: carpal tunnel surgery trigger finger surgery `QUESTION: what can i do to prevent osteoarthritis or its progression `ANSWER: hi i was diagnosed with osteoporosis in 1999 age 53 took 10 mg daily fosamax for ten years next two years i injected my thighs with forteo usually on my right thigh third year injected with reclast then nothing for the fourth year last september i again got injected with reclast from i think around february 2013 i noticed that the joint of my left leg was clicking when i sit for a while i have to hold the top of my thigh to stand up and have to avoid twisting the mri without contrast said i have moderate osteoarthritis in my right hip i read that osteoarthritis is an overgrowth of bone underneath the cartilage can the fosamax forteo and reclast cause this thank you mary t heresa `QUESTION: my sons tibia is fractured and its been a week and a half and he is still having alot of pain is that normal he was fine the past couple of days and now screaming in pain today `ANSWER: can i assume he is in a long leg or short leg cast or is he in a boot any child known to have a tibial fracture one who is screaming in pain needs to be evaluated by his orthopedist as soon as possible this would not be considered normal one and half weeks after the injury `QUESTION: how is osteoarthritis treated what medications can be used `ANSWER: can heavy lifting for years cause did `QUESTION:

```
[ ]: #function to return sample original and generated answers for the validation dataset
def return_validation_sample_output_predicted():
    sample_data=next(iter(validation_gpt_train_dataset))
    v=sample_data[0]['context'].numpy()[0]
    mask_start = len(v) - list(v[::-1]).index(4600) + 1
    input=v[:mask_start+1]
    next_step_input=[]
    input1=list(input)+next_step_input
    final_output=[]
    token_output=[]
    for i in range(0,1024-len(input)):
        input1=list(input1)+next_step_input
        output=tf_gpt2_model([np.array(input1)]['logits'])
        output1=np.argmax(output,axis=1)
        next_step_input=[output1[-1]]#appending last output to input
        next_step_char=tokenizer.decode(next_step_input)
        token_output.append(next_step_input[0])
        final_output.append(next_step_char)
        expected_output=v[mask_start+1:]
    return v,tokenizer.decode(expected_output),tokenizer.decode(token_output)

[ ]: #displaying sample original context, expected answer and generated answer for validation dataset
for i in range(5):
    v,exp,fin=return_validation_sample_output_predicted()
    print(tokenizer.decode(v))
    print(exp)
    print(fin)
    print('-----')
```

to healthcare gov and begin the process `QUESTION: i have medicare now my husband has none how does that work with the new health insurance policies will my medicare change what services and who the doctors it covers how about medicaid `ANSWER: the marketplaces are not for people currently covered by or newly eligible for medicare `QUESTION: what is the official website for obamacare `ANSWER: healthcare gov is the official federal website `QUESTION: has qualifying for medicare or medicaid become easier for those aged 55 65 `ANSWER: you cannot qualify for medicare before the age of 65 `QUESTION: why does health insurance matter for those who are young and healthy `ANSWER: although it is true that young people are healthier on average as a group they do face health problems too about 16 deal with chronic illnesses that require them to seek care on a regular basis and young adults between the ages 19 and 29 visit the emergency room more than any other group under the age of 75 mostly due to accidents what is more there is a significant amount of research that shows young people themselves recognize the value of health insurance to protect them both physically and financially the biggest barrier to health insurance for this group has been cost not a lack of interest in being covered the aca provides financial assistance that is expected to help make insurance much more affordable for the majority of young people entering the market `QUESTION: will having been in a rehab facility for alcoholism in 2013 effect getting insured i have remained sober `ANSWER: starting january 2014 insurers can no longer deny you coverage for a pre existing medical condition including alcoholism substance abuse disorders were included in the ten elements of essential health care benefits covered by the affordable care act aca also known as obamacare meaning all health insurance available on the health insurance exchange marketplace and medicaid insurance has to include services for substance abuse disorders - including alcohol detox www.whitehouse.gov what is more the mental health parity and addiction equity act mhpaa require health insurance companies to provide the same coverage for substance abuse disorders as they do for other medical conditions

Part 3: Deployment using streamlit library in python

```
1 import tensorflow as tf
2 import numpy as np
3 import pandas as pd
4 import streamlit as st
5 import re
6 import os
7 import csv
8 from tqdm import tqdm
9 import faiss
10 from nltk.translate.bleu_score import sentence_bleu
11 from datetime import datetime
12
13 def decontractions(phrase):
14     """decontracted takes text and convert contractions into natural form.
15     ref: https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python/47091490#47091490"""
16     # specific
17     phrase = re.sub(r"won't", "will not", phrase)
18     phrase = re.sub(r"can't", "can not", phrase)
19     phrase = re.sub(r"won't", "will not", phrase)
20     phrase = re.sub(r"can't", "can not", phrase)
21
22     # general
23     phrase = re.sub(r"n't", " not", phrase)
24     phrase = re.sub(r"\'re", " are", phrase)
25     phrase = re.sub(r"\'s", " is", phrase)
26     phrase = re.sub(r"\'d", " would", phrase)
27     phrase = re.sub(r"\'ll", " will", phrase)
28     phrase = re.sub(r"\'t", " not", phrase)
29     phrase = re.sub(r"\'ve", " have", phrase)
30     phrase = re.sub(r"\'m", " am", phrase)
31
32     phrase = re.sub(r"n't", " not", phrase)
33     phrase = re.sub(r"\'re", " are", phrase)
34     phrase = re.sub(r"\'s", " is", phrase)
35     phrase = re.sub(r"\'d", " would", phrase)
36     phrase = re.sub(r"\'ll", " will", phrase)
37     phrase = re.sub(r"\'t", " not", phrase)
38     phrase = re.sub(r"\'ve", " have", phrase)
39     phrase = re.sub(r"\'m", " am", phrase)
40
41     return phrase
42
```

```
59 #importing bert tokenizer and loading the trained question embedding extractor model
60
61 from transformers import AutoTokenizer, TFGPT2Model
62 @st.cache(allow_output_mutation=True)
63 def return_biobert_tokenizer_model():
64     '''returns pretrained biobert tokenizer and question extractor model'''
65     biobert_tokenizer = AutoTokenizer.from_pretrained("cambridgeltl/BioRedditBERT-uncased")
66     question_extractor_model=tf.keras.models.load_model('question_extractor_model_2_11')
67     return biobert_tokenizer,question_extractor_model
68
69
70 #importing gpt2 tokenizer and loading the trained gpt2 model
71 from transformers import GPT2Tokenizer,TFGPT2LMHeadModel
72 @st.cache(allow_output_mutation=True)
73 def return_gpt2_tokenizer_model():
74     '''returns pretrained gpt2 tokenizer and gpt2 model'''
75     gpt2_tokenizer=GPT2Tokenizer.from_pretrained("gpt2")
76     tf_gpt2_model=TFGPT2LMHeadModel.from_pretrained("tf_gpt2_model_2_150_10000")
77     return gpt2_tokenizer,tf_gpt2_model
78
79 #preparing the faiss search
80 qa=pd.read_pickle('./train_gpt_data.pkl')
81 question_bert = qa["Q_FFNN_embeds"].tolist()
82 answer_bert = qa["A_FFNN_embeds"].tolist()
83 question_bert = np.array(question_bert)
84 answer_bert = np.array(answer_bert)
85
86 question_bert = question_bert.astype('float32')
87 answer_bert = answer_bert.astype('float32')
88
89 answer_index = faiss.IndexFlatIP(answer_bert.shape[-1])
90
91 question_index = faiss.IndexFlatIP(question_bert.shape[-1])
92 answer_index.add(answer_bert)
93 question_index.add(question_bert)
94
95
96 print('finished initializing')
```

```

98 #defining function to prepare the data for gpt inference
99 #https://github.com/ash3n/DocProduct
100
101 def preparing_gpt_inference_data(gpt2_tokenizer,question,question_embedding):
102     topk=20
103     scores,indices=answer_index.search(
104         question_embedding.astype('float32'), topk)
105     q_sub=qa.iloc[indices.reshape(20)]
106
107     line = ``QUESTION: %s `ANSWER: ` % (
108         question)
109     encoded_len=len(gpt2_tokenizer.encode(line))
110     for i in q_sub.iterrows():
111         line="QUESTION: %s `ANSWER: %s ` % (i[1]['question'],i[1]['answer']) + line
112         line=line.replace('\n','')
113         encoded_len=len(gpt2_tokenizer.encode(line))
114         if encoded_len>=1024:
115             break
116     return gpt2_tokenizer.encode(line)[-1024:]
117
118
119
120 #function to generate answer given a question and the required answer length
121
122 def give_answer(question,answer_len):
123     preprocessed_question=preprocess(question)
124     question_len=len(preprocessed_question.split(' '))
125     truncated_question=preprocessed_question
126     if question_len>500:
127         truncated_question=' '.join(preprocessed_question.split(' ')[0:500])
128     biobert_tokenizer,question_extractor_model1= return_biobert_tokenizer_model()
129     gpt2_tokenizer,tf_gpt2_model= return_gpt2_tokenizer_model()
130     encoded_question= biobert_tokenizer.encode(truncated_question)
131     max_length=512
132     padded_question=tf.keras.preprocessing.sequence.pad_sequences(
133         [encoded_question], maxlen=max_length, padding='post')
134     question_mask=[[1 if token!=0 else 0 for token in question] for question in padded_question]
135     embeddings=question_extractor_model1({'question':np.array(padded_question),'question_mask':np.array(question_mask)})
136     gpt_input=preparing_gpt_inference_data(gpt2_tokenizer,truncated_question,embeddings.numpy())
137     mask_start = len(gpt_input) - list(gpt_input[::-1]).index(4600) + 1
138     input=gpt_input[:mask_start+1]
139     if len(input)>=1024-answer_len:
140         input=input[-(1024-answer_len):]
141     gpt2_output=gpt2_tokenizer.decode(tf_gpt2_model.generate(input_ids=tf.constant([np.array(input)]),max_length=1024,temperature=0.7)[0])
142     answer=gpt2_output.rindex('`ANSWER: `')
143     return gpt2_output[answer+len('`ANSWER: `'):]
144

```

```

147 #defining the final function to generate answer assuming default answer length to be 20
148 def final_func_1(question):
149     answer_len=25
150     return give_answer(question,answer_len)
151
152
153 def main():
154     st.title('Medical Chatbot')
155     question=st.text_input('Question',"Type Here")
156     result=""
157     if st.button('ask'):
158         #with st.spinner("You Know! an apple a day keeps doctor away!"):
159             start=datetime.now()
160             result=final_func_1(question)
161             end_time =datetime.now()
162             st.success("Here is the answer")
163             st.text(result)
164             st.text("result recieved within "+str((end_time-start).total_seconds()))
165
166
167
168
169
170 if __name__=='__main__':
171     main()

```