AI CHATBOT DESIGN USING DEEP LEARNING BASED NLP

A PROJECT REPORT

Submitted by

JASON PANDIAN

(Reg.No: 714217104011)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

TAMILNADU COLLEGE OF ENGINEERING, COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

MARCH 2021

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"AI Chatbot Design using Deep Learning based NLP"** is the bonafide work of **"Jason Pandian,"** who carried out the project work under my supervision.

SIGNATURE

Dr. K. Krishneswari M.E., Ph.D.,
HEAD OF THE DEPARTMENT,
Department of CSE,
Tamilnadu College of Engineering,
Coimbatore - 641 659

SIGNATURE

Dr. K. Krishneswari M.E., Ph.D.,
SUPERVISOR,
Professor and Head,
Department of CSE,
Tamilnadu College of Engineering,
Coimbatore - 641 659

Submitted for the Anna university examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Making a machine to think and act as a human has always been a wild imagination of mankind for centuries. From the perspective of computing, Information Retrieval (IR) and Natural Language Processing (NLP) are the key technologies to address this issue to make our dreams come alive. Most of the human advances of this century were only due to the development of communication technologies that were very much rely on IR and NLP. Nowadays, Natural Language Processing (NLP) plays a tremendous role in human life. In this project, a Chatbot design was built using Deep Learning(DL) technologies with the help of cloud platform. It also produced fine results as like other proprietary Chatbots. The proposed Chatbot can handle open domain as well domain specific questions.

DEDICATION

I dedicate this dissertation work to my family and friends. A special feeling of gratitude to my loving parents, Mr. Pandian and Mrs. Shanthi, whose words of encouragement and push for tenacity ring in my ears. My sister Miss. Jessie Rithika and my grandmother Mrs. Kamalam Chandrabai have never left my side and are very special. I will always appreciate all they have done, especially my father, for helping me develop my technology skills. I dedicate this work and give special thanks to all the great and generous humans of open source researchers and GNU/Linux community.

ACKNOWLEDGEMENTS

I express my deep gratitude to the management for providing a good infrastructure to do this project.

I express my thanks to our Principal **Dr. M. KARTHIKEYAN M.Tech., Ph.d.,** for his continuous support in all our curricular activities.

I am thankful to my project supervisor **Dr. K. KRISHNESWARI M.E., Ph.D.,** Head of the Department for her guidance, encouragement, valuable suggestions and feedback for completing the project in good shape.

I thank the project coordinator **Mr. R. SELVARAJ M.E.,** for leading me to complete this project in time.

I extend my sincere thanks to all the teaching and nonteaching faculty members of the department for their assistance and all our friends who helped us in bringing out our project in good shape and form.

I express my gratitude to my family for their encouragement and support in all my endeavours.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI Artificial Intelligence.

ALBERT Lite Bidirectional Encoder Representations from Transformers.

ANN Artificial Neural Network.

BERT Bidirectional Encoder Representations from Transformers.

BPN Backpropagation Network.

CNN Convolutional neural network.

DistilBERT a distilled version of BERT.

DL Deep Learning.

DMN Dynamic Memory Network.

DNC Differentiable Neural Computers.

DNN Deep Neural Network.

GPU Graphical Processing Unit.

IR Information Retrieval.

ML Machine Learning.

MM Memory Network.

MT Machine Translation.

NLP Natural Language Processing.

NMT Neural Machine Translation.

PyPI Python Package Index.

QA Question and Answer.

QAS Question Answering System.

RAM Random Access Memory.

RNN Recurrent Neural Network.

RoBERTa A Robustly Optimized BERT Pretraining Approach.

SQuAD Stanford Question Answering Dataset.

TPU Tensor Processing Unit.

CHAPTER 1

INTRODUCTION

Making a machine to think and act as a human has always been a wild imagination of mankind for centuries. Information Retrieval (IR) and Natural Language Processing (NLP) are the key technologies to address this issue to make our dreams come alive. Most of the human advances of this century were only due to the development of communication technologies that were very much relied on IR and NLP. Nowadays, Natural Language Processing (NLP) plays a tremendous role in human life.

Chatbots have recently become a significant application in industry. People of this decade use web-based technology more than ever before. In the business part, the companies and organizations use skilled persons to keep communicate with their customers and solve their queries by feedback. As it was literally involved by humans effort. In organizations, skilled persons spend their time by answering a lot of questions to their customers. To avoid those problems and reduce human involvement and time, a pretrained chat-

bot can be used. The chatbot system is cost-efficient and it will need no rest and retirement. In this project, we will address a hybrid model of a chatbot using BERT based deep learning network that can respond to either Domain-specific or Open-domain questions.

1.1 Introduction

It will be difficult for the user to achieve particular information directly from search engines, where they got many results to parse manually. Getting specific information about a product from an organization take a long time by normal human to human conversation and even while browsing website of the organization or information booklets about an organization. Information Retrieval (IR) and Natural Language Processing (NLP) are the important technologies dealing with this problem.

1.1.1 Motivation

Most of the human advancements of this century were only because of the development in communication technologies. IR and NLP based technologies. Most of available solutions are proprietary technologies so one can not get open access to understand the internal mechanisms of such NLP technologies.This motivates me to do a custom AI Chatbot using Deep learning based NLP technologies. Furthermore, in this decade, there is a lot of research activity in NLP technologies because of the recent growth in AI and deep learning networks. Since NLP is the future of internet and human communication, it is motivating me do take a small step in that direction.

1.1.2 Related Works

In [16], Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova from Google AI Language introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations of Transformers. The pretrained BERT model can be fine- tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT obtains new state-of-the-art results on natural language processing tasks. This model could not handle large text blocks during training and testing. So a chatbot using this model can not handle large text blocks for training and testing.

Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length. To address this limitation, in [15] Iz Beltagy, Matthew E. Peters, and Arman Cohan introduce the Longformer with an attention mechanism that scales linearly with sequence length, making it easy to process documents of thousands of tokens or longer. This model can accommodate large text blocks during training and testing in a very memory efficient manner. So this model can be adopted to implement a good AI chatbot.

In [3], Amit Mishra and Sanjay Kumar Jain surveyed QASs and classifies them based on different criteria. They identify the current status of the research in each category of QASs, and suggest the future scope of the research.

In [25], Mohammad Nuruzzaman and Omar Khadeer Hussain conducted an in-depth survey of recent literature, examining over 70 publications related to chatbots published in the last 5 years (before 2019). Based on that literature review, that study made a comparison with selected papers accord-

ing to method adopted. This paper also presented why current chatbot models fail to take into account when generating responses and how this affects the quality conversation.

1.1.3 Disadvantages of Previous System

Classical chatbot designs that are based on early NLP techniques are not efficient. They can not provide a specific answer to a question. Generally, they are implemented for a specific task with limited domain knowledge and can not provide answers to Open-domain questions. Since early QA Chotbot models may use old machine learning algorithms, they can not handle complex QA tasks.

1.1.4 Proposed System & its Advantages

It will provide appropriate and selective answers from the web so the user need not filter or parse bulky search results. The system can handle Open-domain questions. So it can predict answer almost every time. The Open domain search was fully focused on client privacy because it uses DuckDuckgo and Wikipedia APIs for search. It Can be easily implemented and customized for a specific task in a specific application domain It is cost-effective and well suited to work in every browser. The system can answer user queries with chit-chat answers, which reduces traditional knowledge gathering/learning time. The system uses DL based NLP technology to minimize the bot misunderstanding so that the answers will be more specific than traditional systems.

1.1.5 Project Objectives

The objective of the project is to create a question answering system that can answer the queries based on the DuckDuckgGo search engine and Wikipedia and also provide a personalized chatbot for a particular organization to provide a quick, reliable and direct answer for customers questions from an information base.

1.1.6 General Constrains of the Project

Our implementation of this Chatbot supports the English language only. Generally, a Chatbot can not hold the previous conversation held with the user to understand the context of the discussion. But in our implementation, we will instruct the chatbot to set the context before the QA session. In our implementation, we use remote cloud resources for computation. So that this implementation of Chatbot consumes time while processing some kind of responses. In our implementation, Chatbot can't perform math operations.

1.1.7 Outline of this Thesis

This chapter provided a minimal introduction to the project. The next chapter will present a survey on natural language processing systems .The chapter 3 will explain about the study and evaluation on deep neural networks in detail. The chapter 4 will present design of chatbot system. The chapter 5 will present the architecture of the proposed chatbot. The chapter 6 will present about the implementation, results and discussion. Finally, chapter 7 will conclude with a brief conclusion and future scope.

CHAPTER 2

SURVEY ON NATURAL LANGUAGE PROCESSING SYSTEMS

In this survey, we tried to present a brief overview of 80 years of NLP history. Even though we outlined some of the classical as well as modern NLP methods in this survey, we tried to give special attention to some of the state-of-the-art Deep Learning(DL) based NLP models that are giving new perspectives for Artificial Intelligence(AI) and other related technologies.

2.1 Introduction

Deep learning models can be fit into various application domains and could produce state-of-the-art results. In recent years, Deep Learning models became a great tool for solving different kinds of natural languages problems. In this review, to provide a clear walk-through of deep learning-based NLP models and their evolution, we summarize, compare, visualize, and highlight

the various NLP methods as well as deep neural network models. We review deep learning models that can solve real-time problems and can be employed for numerous NLP tasks and try to understand the past, present, and future trends of deep learning-based NLP systems.

2.1.1 Deep Learning (DL)

DL models can recognize and classify complex patterns in data using a deep, multilayered networks paradigm. It is often used for machine learning and artificial intelligence-related tasks. Some of the examples of deep learning algorithms are Deep Boltzmann Machine (DBM), Deep Belief Networks (DBN), and Convolutional Neural Networks (CNN). Generally, NLP models were designed using the CNN algorithm, so this project will only address CNN related models.

2.1.2 Natural Language Processing(NLP)

NLP is one of the interesting and useful applications of Artificial Intelligence. In simple terms, an NLP system will try to mimic the ways of the human language understanding model to make sense of textual or speech content. Generally, this kind of NLP tasks will require the system to have the capability to translate, analyze, and synthesize the language to make sense out of it.

Humans can effortlessly process both textual or speech content and understand them. The main task of an artificial NLP system is to replace the human aspect of understanding with a machine aspect of understanding and make the machine to understand the natural language input and act accordingly. Mimicking the process of Understanding natural languages will be difficult to develop because of the numerous ambiguities and levels of con-

textual meaning involved in natural language. The inherent ambiguity of a language makes the NLP difficult to understand from a machine's perspective. There are five main categories into which language ambiguities fall: syntactic, lexical, semantic, referential, and pragmatic[10]. These five aspects of language make it difficult to design a machine to act as a human to understand language content.

2.2 A Brief History of NLP

In [21], Karen Sparck Jones did a detailed review of NLP techniques from their origin in the late 1940s to the early 1990s. Karen's review identified and clearly distinguished four phases of NLP history based on the characteristics namely 1) emphasis on machine translation, 2) by the influence of artificial intelligence, 3) by the adoption of a logico-grammatical style, and 4) by an attack on massive language data. She defined the first phase of work in NLP as lasting from the late 1940s to the late 1960s, the second from the late 60s to the late 70s and the third from late 70s to the late 80s, and the fourth phase from the late 80s to the late 90s.

As shown in the previous paragraph, Karen's survey covered and classified almost 40 to 50 years of NLP history. Our survey extends her work by adding a fifth phase of NLP history which covers the remaining 30 years of NLP history until now; particularly, the deep learning related ones.

2.2.1 Phase I: Late 1940s to Late 1960s

Mostly the work of the first phase was focused on machine translation(MT). The following are some of the interesting aspects of the first phase pointed out by Karen.

- A MT task was done by translation by lookup using dictionary-based word-for-word processing.

- The computing resources available at that early period were very limited. The notable characteristics of that time were: The era of punched cards and batch processing. There were no suitable higher-level languages and programming was virtually all in assembly language. Access to computing machines was often restricted; they had very finite storage and were extremely slow.

- Even with the most advanced algorithms running on the best machines available at that time, for analysing long sentences, takes several minutes of processing time was needed which makes it impractical for implementing real-time MT systems.

- The practice of using computers for literacy and linguistic study began in this period, but none of them was closely linked with NLP at that time.

This phase was about "syntax and semantics-based". To make formal theories and concepts fit into hardware researchers were seeking for a novel computing hardware for data processing the NLP tasks. However, data processing itself was not well established in this early period of time.

2.2.2 Phase II: Late 1960s to Late 1970s

The second phase of NLP work was artificial intelligence (AI) flavoured one. The following are some of the interesting aspects of the second phase pointed out by Karen.

- It gave much more emphasis on world knowledge and on its role in the construction and manipulation of meaning representation.
- Mostly, the actual input to these systems was restricted and the language processing involved very simple compared with contemporary MT analysis.
- The latter works realized the need for inference on the knowledge base in interpreting and responding to language input.
- The need to identify the language user's goals and plans was early recognised and has become a major trend in NLP research since, along with a more careful treatment of speech acts.

This phase was AI-flavoured and semantics-oriented. The identification of language user's goals and plans become the major trend in this period of time.

2.2.3 Phase III: Late 1970s to Late 1980s

The third phase can be described as a grammatico-logical phase. The following are some of the interesting aspects of the third phase pointed out by Karen.

- This trend, as a response to the failures of practical system building, was stimulated by the development of grammatical theory among linguists during the 70s, and by the move towards the use of logic for knowledge representation and reasoning in AI.
- Computational grammar theory became a very active area of research linked with work on logics for meaning and knowledge representation

that can deal with the language user's beliefs and intentions and can capture discourse features and functions like emphasis and theme, as well as indicate semantic case roles.

- The grammatico-logical approach was also influential in some other ways. It led to the widespread use of predicate calculus-style meaning representations, even where the processes delivering these were more informal than the purist would wish.

- The fourth trend of the 80s was a marked growth of work on the lexicon.

This phase was grammatico-logical. It satisfies the language user's beliefs and goals.

2.2.4 Phase IV: Late 1980s Onward

This phase was labelled as a massive data-bashing period and the following are some of the interesting aspects of the fourth phase pointed out by Karen.

- The rapid growth in the supply of machine-readable text has not only supplied NLP researchers with a source of data and a testbed for e.g., parsers.

- Use of text retrieval (cf. Jacobs, 1992) for document retrieval made this phase possible to process and available to various domains.

- This phase also concentrate on open question problems by the development of realistic NLP techniques like document retrieval and text parsers etc.

- This phase includes all the previous ideas of the past three phases: from syntax to semantics and semantics to grammatico-logical phases and grammatico-logical phases to lexicon(now).

- The development of some good processing systems as discussed in the third phase made user life easier. The users of this era benefited from the development of GUI for user experience. Hence, it was the major drawback of the last three phases.

By considering the negatives of the past three phases. This period has seen a significant, new interest in multi-modal, or multimedia systems. But whether combining language with other modes of media, like graphics, actually simplifies or complicates language processing was an open question until this fourth phase.

2.2.5 Phase V: Late 1990s to the Deep Learning Era

Even though the history of Deep Learning[22] can be traced back to 1943, the significant evolutionary step for DL took place only in 1999, when computers started becoming faster at processing data and graphics processing units (GPU) were developed[22].

Some of the important factors influencing the development of practical deep neural networks are :

- The developments towards Deep Learning was possible only because of the increase in memory, storage and computing capabilities of modern computers.

- It become possible because of the improvements in the capabilities of the Graphical Processing Unit (GPU)[22].

- It become possible because of the invention of Tensor Processing Units (TPUs)
- The research in this area developed very fast because of the open nature of research in DNN related technologies
- Another major factor that influenced this development was the ”free” cloud computing resources generously provided by Google - this only promoted this research to a higher level

The following are the Noticeable milestones of DNN based models :

In 2001, a research report by META Group (now called Gartner)[22] described the increasing volume of data and the boosting speed of data as boosting the range of data sources and types. This was a call to prepare for the aggression of Big Data, which was just starting. This year is to be considered as the rise of Natural language models.

In 2008, two authors Collobert and Weston proposed a work on Multi-task learning[6]. It has been used successfully across all applications of machine learning from natural language processing. Collobert's and Weston's work was an eye-opener for speech recognition to computer vision.

In 2011, the acceleration of GPUs had increased significantly, making it possible to train convolutional neural networks "without" the layer-by-layer pre-training[22]. With the increased computing speed, it became clear Deep Learning had significant advantages in terms of efficiency and speed.

In 2013, Tomas Mikolov, et al.[31] from Google created a word embedding toolkit called as Word2vec. It can train vector space models faster than the previous approaches. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace. So it was considered as an eye-opener for novel DNN based NLP models.

In 2014, Ilya Sutskever, et al.[13] from Google created a deep neural network called as Sequence to sequence learning with neural networks. It was a Long short term memory(LSTM) model, that aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ. So it performed well in language-based tasks.

In 2015, Dzmitry Bahdanau, et al.[9] proposed a machine translation based model, that is one of the core innovations in neural MT (NMT) and the root idea that enabled NMT models to outperform classic phrase-based MT systems and it was the best model to perform English to French Translation at that time.

In the same year 2015, the researchers came in different variants of approaches such as

- Alex Graves, Greg Wayne and Ivo Danihelka[1] created a deep neural network model called Neural Turing Machine. It was developed with a working memory, which gives it very impressive learning abilities. Unlike a standard neural network, it also interacts with a memory matrix using selective read and write operations.

- Jason Weston et al.[20] created a deep network model called as Memory network, It combines learning strategies from the machine learning literature with a memory component that can be read and write.

- Sainbayar Sukhbaatar et al.[28] created a deep network model called End-to-end Memory Networks. It was developed with a recurrent attention model over a possibly large external memory. The architecture is a form of Memory Network, but it is trained end-to-end, and hence requires significantly less supervision during training.

- Kumar et al.[4] created a deep network model called Dynamic Mem-

ory Network(DMN). It processes input sequences and questions, forms episodic memories, and generates relevant answers. Due to its dynamic architecture, it was mainly optimised for question-answering(Q&A) problems and becomes a new path for NLP.

- Graves et al.[2] created a novel model called differentiable neural computers (DNCs). DNCs was typically recurrent structure in its implementation, but it can learn like standard neural networks, but that can also store complex data like computers. DNCs indirectly takes inspiration from Von-Neumann architecture, making it likely to outperform conventional architectures in tasks.

- Henaff et al. created a deep network model called Recurrent Entity Network (EntNet). EntNet was equipped with a dynamic long-term memory which allows it to maintain and update a representation of the state of the world as it receives new data.

The above mentioned models help the DL-based NLP community to be developed in a wide area.

In 2016, Pranav Rajpurkar, et al. of Stanford University prepared a Q&A dataset. They call their work[8] as SQuAD and it has organized more than 100,000 questions from Wikipedia articles. Using SQuAD, they evaluated the human performance of this QA task and posted it on the SQuAD website(with leader board)[29]. In SQuAD leader board, they posted their validated human performance on SQuAD in terms of F1-Score was 91.221.

In the same year, the Singapore Management University created a Long Short term memory network model and achieved an F1 score of 67.748[29], which was below human-level performance.

In 2017, the CMU created a model called Conductor-net and achieved an F1 score of 81.415[29], it was a fine improvement within a year, and also

it was near human-level performance. In the same year, Ashish Vaswani et al. at Google Brain proposed a deep transformer model based on attention mechanism[5]. It entirely changed the views of the researchers about the NLP. In the year 2018, Google AI Language released a deep bi-transformer model based on attention mechanism called BERT[16]. It became famous and productive for different NLP tasks. In the year 2019, Facebook AI research published another BERT based model called RoBERTa[34]. It achieved an F1 score of 89.795 from [30]. This was near equal human-level performance. In the year 2020, another enhanced Albert based model named QIANXIN's SA-Net achieved an F1 score of 93.011[30]. This model outperformed humans.

Currently, in this decade, the processing of Big Data and the evolution of Artificial Intelligence are both dependent on Deep Neural Networks and Deep Learning techniques.

2.2.6 Observations and Findings

As we did this survey, we realized the important improvements in NLP techniques of the different phases of NLP history. Based on that observation, with respect to performance, we can classify all algorithms of NLP into five classes as algorithms that are giving:

- Below Human-level Performance
- Near Human-level Performance
- Equal Human-level Performance
- Above Human-level Performance and
- Super/Hyper Human level of Performance

If we carefully study the five phases of NLP presented in our Brief History of NLP in section 2, it is obvious that almost all algorithms or models invented up to the first four phases of NLP history were only achieved 'Below Human-level Performance'. Only during the fifth phase of NLP history, the algorithms and models achieved 'Near Human-level Performance', 'Equal Human-level Performance', and even 'Above Human-level Performance'. And DNNs play an important role in achieving these levels of performance.

2.3 Summary

As we observed in our survey, DL based NLP models performed poorly even until 2017 and their results are very lower than human-level performance. After that, with the availability of heavily powered machines, DL based NLP models started to perform good, but their performance was still a little bit low comparing with actual human performance. During 2018, some state-of-the-art models evolved and played a huge role in NLP tasks and achieved equal to human performance. After that, Multi powered(layered or ensembled) state-of-the-art models evolved and started to achieve above-human level performance in various NLP tasks. As we mentioned in the previous section, the NLP model based on Deep Learning is quite interesting and was able to achieve an above-human level of performance.

CHAPTER 3

STUDY AND EVALUATION ON DEEP NEURAL NETWORKS

3.1 Introduction to DNN

3.1.1 Machine Learning

Machine learning is a subfield of Artificial Intelligence that can process data and fit that data into a model and can be used almost in every field. Even though machine learning comes under computer science, it is very much different from traditional computational techniques. In the classical computing model, algorithms are sets of programmed instructions that computers can solve with respect to their computational power. Basically, machine learning algorithms make a model to train data inputs and use the trained model to produce output within a specific range. Thus, machine learning helps to build models from sample data in order to automate decision-making based

on data inputs. People today directly or indirectly enjoying the power of machine learning in one way or another.

3.1.2 Artificial Neural Networks (ANN)

Traditional ANNs or simply NNs are simple mathematical structures generally denoting neurons with less number of hidden states.

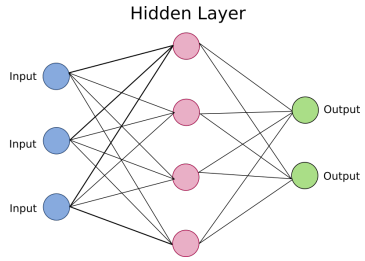

Figure 3.1: Block Diagram of Traditional Neural Networks.

The first traditional neural network or artificial neural network(ANN) was a Perceptron network created by Rosenblatt in 1958 and used pattern recognition with mathematical notation[27]. In 1974, Another model was proposed by Seppo Linnainmaa[23]. That model evaluates the gradient of the loss function with respect to the weights of the network for a single input-output. The idea of BPN was raised by various scientist in the early 60's. But Linnainmaa was the first to propose and implemented to run on computers, that this approach could be used for the neural network after analyzing it in-depth in his 1974 dissertation. In the history of ANN, this same ideology with a change in the model may result in different NN architecture with respect to the computational power of the systems. Hence, these ANN can process entry-level tasks. When it comes to classical problem solving. In the past, Perceptron network and BPN were used to solve different types of image processing and classification tasks.

3.1.3 Deep Neural Networks(DNN)

DNN was inspired by the human brain and tries to mimic the functions of the human brain. A DNN is a network with a large number of hidden states. These hidden states (layers of neurons) can be used to process multidimensional inputs and can be tuned for complex classification tasks.

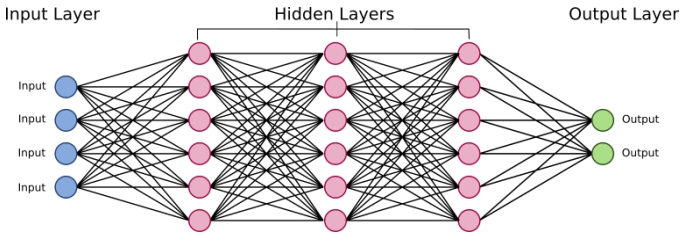

Figure 3.2: Block Diagram of Deep Neural Networks.

3.1.4 Recurrent Neural Network(RNN)

It was introduced in the 20th century by Psychologist David and Rumelhart. According to their work[33], the human biological brain functions the sequential data(biological data) by memorizing all past sequences. This principle is applied to the artificial neural network to memorize the data, the neural network is designed with the initial memory to process a sequential data. In [19], the word “recurrent ” means repetition, by processing the sequential data recurrently and storing its past sequences makes RNN works like a looped(feedback) network. Thus RNN can make an accurate prediction with temporal data-based applications.

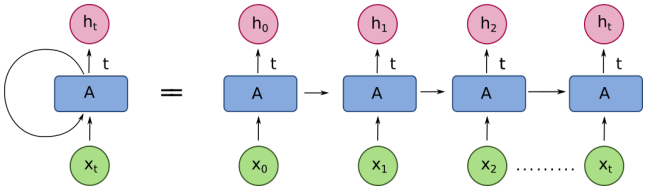

Figure 3.3: RNN Folded and Unfolded Version.

An RNN has four blocks,

- x(t) – input block, which gets the sequential data.
- A – the main block that contains the weight and activation function of the network.
- t – time-step, that holds the time required to process each sequence.
- h(t) – output block, the present output will be based on the previous outputs.

RNN has been the basic approach for training the sequential data and it has been the key to the development of new powerful models like LSTM and Gated Reccurent Unit(GRU) networks. The vanishing gradient problem was the major problem in RNN. From the perspective of NLP, an RNN can handle only small text sequences(RNN has short-term memory). It is difficult to train long sequences using normal RNN and also RNN may not be used for language translation. Further, training an RNN requires a large amount of cost and time. RNN is the foundation of different NLP models which are available today.

3.1.5 Convolutional Neural Network (ConvNet/CNN)

Biological Perspective of CNN

In [7], Hubel and Wiesel in the years 1950s and 1960s proved that cat and monkey's visual cortexes contain neurons, that individually respond to small regions of the visual field without the movement of the eyes. The region of the visual space within which visual stimuli affect the firing of a single neuron is known as its receptive field. The Connecting cells have similar and overlapping receptive fields. The Receptive field size and location differs systematically across the cortex to form a complete map of visual space. The

cortex in each hemisphere describes the contralateral visual field. Their work analyzes two basic visual cell types in the brain:

- Simple cells, whose output is maximized by straight edges having particular orientations within their receptive field.
- Complex cells, which have larger receptive fields, whose output is non-responsive to the exact position of the edges in the field.

They proposed a cascading model of these two types of cells for using it in pattern recognition tasks.

In 1980, Kunihiko Fukushima introduced ”neocognitron”[11, 12]. His work explains about the architecture of CNN. Even though it architecture was originally inspired by Hubel and Wiesel, which was analogous to that of the connectivity pattern of neurons in the visual cortex of the human brain. Individual neurons respond to stimuli only in a closed region of the visual field known as the receptive field. A collection of receptive field overlaps to cover the entire visual area. This principle used to develop CNN architecture.

Design of a CNN

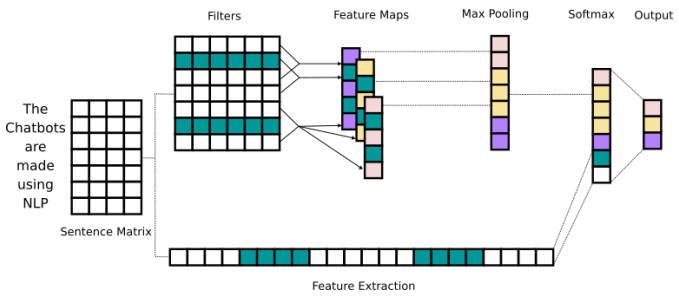

Figure 3.4: A CNN based NLP Task.

A CNN is a feed-forward deep learning network that passes inputs only in one direction, forward, from the first convolution layer, through all other convolution layers, and to the fully connected layer, and output is obtained

expresses various aspects of the input[18]. A convolution is a process that transforms two functions into a new function. The pre-processing required in CNN is much lower than classical classification algorithms. While in primitive methods, filters are hand-engineered. On the other hand, CNN has the ability to learn these filters/characteristics by itself.

That means CNN needs no manual pre-processing. The CNN filters are not defined instead the value of every filter is learned during the training process itself. CNN executes hierarchical feature learning. CNN does not encode the position and orientation of objects. This makes convolution operations well carried out in low-end machines. Such as smartphones, embedded systems, and IoT devices. The modern CNN is one of the main categories to do image recognition, image classification, as well as considered the key for the development of almost all the novel NLP models that are good in running real-time applications that are available today.

3.1.6 Difference Between Neural Networks and Deep Neural Networks

Deep Neural Networks is nothing but a very large traditional NN, which will typically need higher-end hardware to make real use of it. In Table 3.1, we present some of the differences between traditional neural networks and deep neural networks.

3.2 Deep-Learning-Based NLP Systems

Deep Learning is not a novel field. The first Deep Learning model was published by Alexey Ivakhnenko and Lapa in 1967. Paper[14] described a deep network with eight layers trained by the group method of data handling,

Table 3.1: Difference Between NN and DNN [17]

	Traditional Neural Networks (NN)	**Deep Neural Networks (DNN)**
Hardware	Traditional Neural Networks require only Lower-end processors (i.e, it can function with less number of Central Processing Unit Cores) and Lower-end Graphics Processing Units. It also requires less power supply	DNN requires Higher-end processors (ie, it can function only with a high number of Central Processing Unit Cores), Higher-end Graphics Processing Unit and Tensor Processing Unit. It also requires a large amount of power supply.
Training Time	Traditional NN can be easily trained and executed in low-end machines. It requires less amount of time to train a typical neural network	DNN can only be trained on higher-end hardware and will require several hours or even several days of CPU/GPU/TPU time.
Training Algorithm	Traditional Simple training algorithms such as SGD will be sufficient to train a Traditional NN for a typical problem.	The complexity and vastness of DNN problem space will require the best algorithms (Adam and its variants are believed to be providing better performance on DNN.) Block Diagram of Traditional Neural Networks
Epochs	A small problem related to Traditional NN typically will require several epochs for achieving better performance. Further improvement of training will linearly increase with respect to the training epochs. So hopefully we will have a better-trained model at the final epoch.	In most of the NLP problems, generally, the performance of the training will not get improved with respect to the epochs as the case of NN. However, it will achieve acceptable accuracy even in a few epochs. However, most of the time, the model belonging to the final epoch will not be the better model. Even a model belonging to a lesser epoch may perform well. So as a practice, people will choose a better model from N epochs of training.
Parameters	A typical Traditional NN and a related problem will generally need less than a hundred or a few hundred parameters during training.	On the other hand, a typical DNN generally needs millions of parameters for solving problems such as NLP Applications.
Example	SVM, RBF, etc., are the most successful Traditional NN models	There are a lot of Successful models with respect to the area of their application.(LSTM, Bi-LSTM, CNN, BERT, You only look once(YOLO), etc)

while the algorithm worked, training required by this algorithm took 3 days. It is clear that Deep Learning requires high computation power. By definition, it is a family member of machine learning that can automatically extract the features from the data while processing large datasets. It was often interchanged with the word deep neural networks or artificial neural network. Hence, nothing was changed comparing late 60s and 70s, except the extreme computation power and parallelization techniques of 21st-century computing machines.

In this section, we present some of the popular and most widely used deep learning models of recent years. The models described here are widely used in the design of NLP systems. So these models are called as deep language models. We also focus these models based question answering systems(QAS). The one commonality in all the discussed models is: all of them are 'transformers' based on attention mechanism.

3.2.1 Bidirectional Encoder Representations from Transformers (BERT)

BERT was a deep neural network(language model) created by Jacob Devlin, et al. from Google[16]. It was made up of transformers, and stack multiple transformer encoders on top of each. It uses bidirectional learning as opposed to directional models. BERT tries to understand the whole context by addressing each word from left and right. BERT architecture is based on Self-Head Attention. BERT uses two pre-training strategies: masked language modelling(MLM) and next sentence prediction (NSP). BERT was readily pretrained using these two pre-training strategies on the BookCorpus (800 million words) and English Wikipedia (2500 million words). This makes pretrained BERT as distinguished from past DNN models. It can readily

used in medium cost machines. By transfer learning, BERT can be modified into various NLP tasks by adding one or a few core layers in its end. This process is called as fine-tuning. These salient features of the BERT made NLP enthusiasts to personalize, develop, evolve, research and optimize new models.

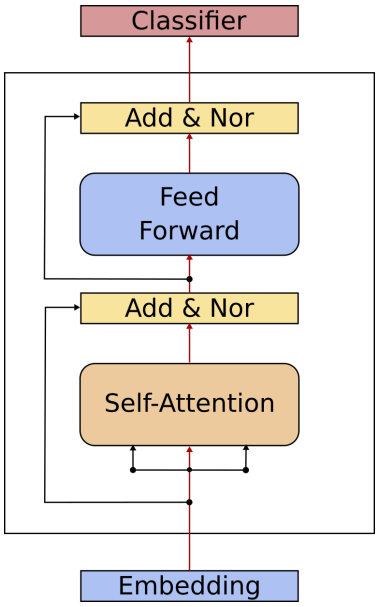

Figure 3.5: BERT Architecture with self-attention.

The“Fig. 3.5” shows the architecture of BERT. As shown in this figure, the self-attention block was responsible for possible word prediction that was described in [16]. Other blocks are common functional blocks found almost in every transformer-based architectures.

3.2.2 XLNet

XLNet was a deep neural network(language model) created by Zhilin Yang et al. from Google AI Brain Team[36]. It was a large bidirectional transformer, that was improved the training process with larger data and use more computational power to achieve better than BERT prediction metrics on 20 language tasks. To improve the training, It introduces permutation

language modelling, where all tokens are predicted but in random order. Its architecture was based on Two-stream Self-attention. This contrasts with BERT's masked language model with self-attention, where only the masked (15%) tokens are predicted. This was also in contrast to the other language models, where all tokens were predicted in sequential order instead of random order. This helps the model to learn efficient bidirectional relationships and therefore better handle the dependencies and relations between words. In inclusion, Transformer XL was used as the base architecture, which produced good performance even in the absence of permutation-based training.

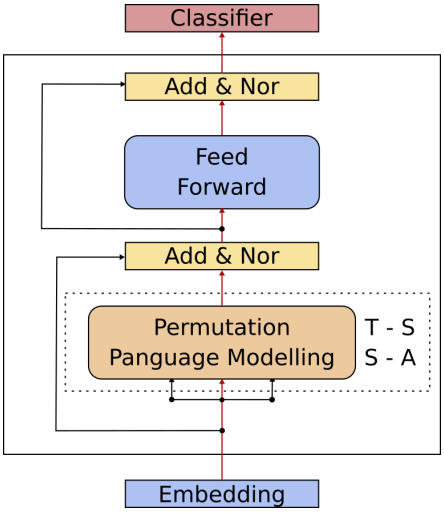

Figure 3.6: XLNet Architecture with Two-stream Self-attention

The "Fig. 3.6" shows the architecture of XLNet, As shown in this figure, the permutation language modelling block was responsible for random order predictions that were described in [36]. Other blocks are common functional blocks found almost in every transformer-based architectures.

3.2.3 Robustly-optimized BERT approach (RoBERTa)

RoBERTa was a deep neural network(language model) created by Yinhan Liu et al. of Facebook[34]. It outperformed both XLNET and BERT in the Glue benchmark. It improved the BERT masked language modelling with

a strategy where they remove Next Sentence Prediction in BERT and introduced an idea called dynamic masking. Hence that masked token changes throughout each epoch, which made the model to learn to predict intentionally hidden secrets of text. They also improved the hyper-parameters tuning for the BERT and trained the BERT model using much larger mini-batches and learning rates.

The“Fig. 3.7” shows the architecture of Roberta. As shown in this figure, the Dynamic modelling and Full Sentence without NSP(DyM & Full sen without NSP) block is responsible for possible word predictions that were described in [34]. Other blocks are common functional blocks found almost in every transformer-based architectures.

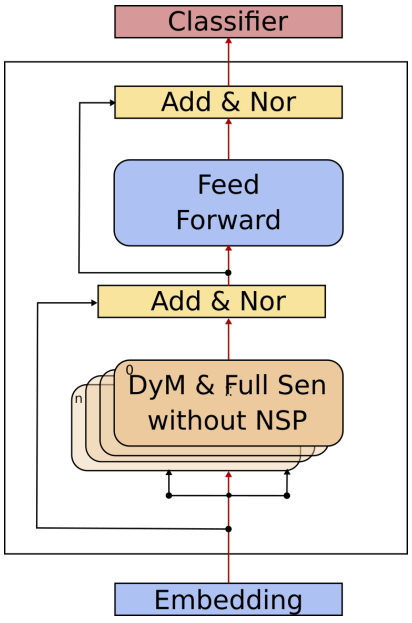

Figure 3.7: RoBERTa Architecture with Dynamic Modeling.

3.2.4 DistilBERT

DistilBERT was a deep neural network(language model) created by the NLP enthusiasts Victor SANH et al. at Hugging Face[32]. It showed that it is possible to reach and achieve 97% of BERT’s language understanding

capabilities while reducing the size of the BERT model by 40%. Moreover, this model was 60% faster. While relying on the BERT architecture and using the same training data, DistilBERT removed the token-type embeddings and pooler (which BERT uses for the next sentence classification task) and also implemented a few ideas from RoBERTa and used a knowledge distillation process for the training of the model.

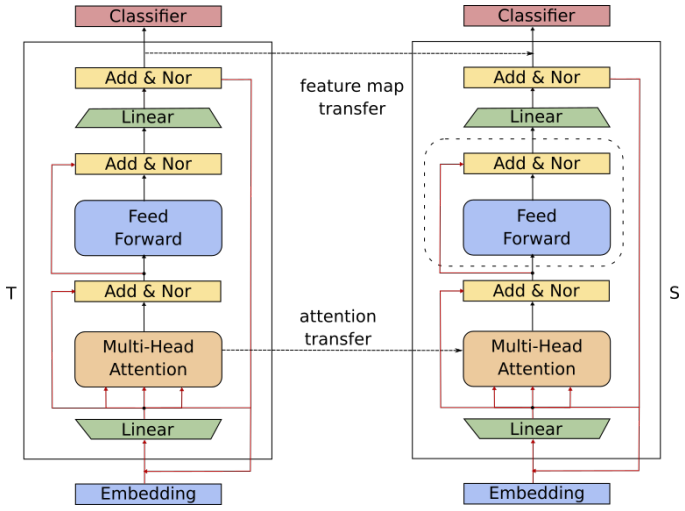

Figure 3.8: The Teacher-Student Architecture of DistilBERT

The“Fig. 3.8” shows the architecture of DistilBERT. As shown in this figure, the T & S describes the teacher-student model, DistilBERT - student(S) can perform up to 97%n of the knowledge gained from the 100%n BERT - teacher(t), even student was reduced into 40%, it can perform 60% faster. So DistilBERT can be used for low cost based NLP tasks.

The term “DistilBERT” is derived from the distillation process from science, which is a process of separating water and salt. In NLP, distillation refers to knowledge distillation which means to train a student model (DistilBERT) based on an already trained teacher model (BERT). In this case, DistilBERT is trained to copy the behaviour of BERT by equaling the output distribution - training through knowledge transfer.

3.2.5 A Lite BERT (ALBERT)

ALBERT was a deep neural network(language model) created by Zhenzhong Lan1 Mingda Chen et al. from Google Research collaboration with Toyota Technological Institute[35]. They pointed out that, the RoBERTa focused on performance and DistilBERT on speed, ALBERT is built to address both. Their model achieved better results with lower memory consumption and increased training speed compared to BERT. They also claim that BERT was parameter inefficient and apply techniques to reduce the parameters to 1/10th of the original model without substantial performance loss. Building on the BERT architecture, they demonstrate two strategies to reduce the model size: factorized embedding parameterization and cross-layer parameter sharing. In inclusion, they improved the model training by changing the next-sentence-prediction task for sentence-order prediction by keeping the equal amount of training data as BERT and DistilBERT.

The “Fig. 3.9” shows the architecture of ALBERT. As shown in this figure, the “self-supervised loss for Sentence-order Prediction(SOP) block” was responsible for possible word predictions that were described in [35]. Other blocks are common functional blocks found almost in every transformer-based architectures.

3.3 Comparison of Different DNN Models and Their Performance

In this section, we present some of the important parameters and the performance scores of five deep neural network models, BERT, XL-Net, RoBERT, DistilBERT and ALBERT. The results of the performance were arrived on the popular Question Answering Dataset called SQuAD.

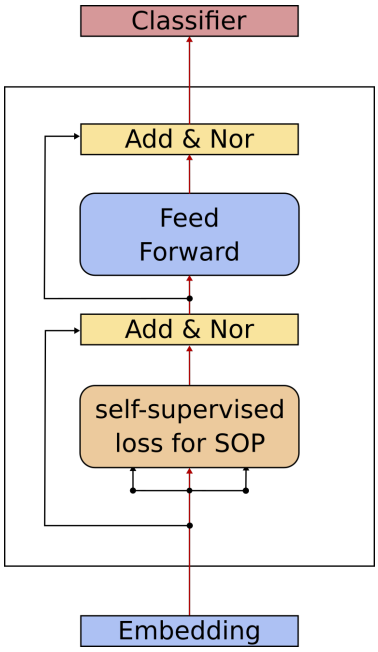

Figure 3.9: ALBERT Architecture with SOP

3.3.1 Stanford Question Answering Dataset (SQuAD)

SQuAD1

The SQuAD was published in 2016 by Pranav Rajpurkar et al. from the Stanford proposed dataset called SQuAD[26]. It was a reading comprehension dataset, consisting of 100,000+ questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of the text or span from the corresponding reading passage, or the question might be unanswerable. After the development of SQuAD 2.0, the SQuAD was named as SQuAD1.1.

SQuAD2.0

SQuAD2.0 was also published in 2018 by Pranav Rajpurkar et al.[8]. It combines the 100,000+ questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. To do well with SQuAD2.0, systems must not only answer

questions when possible but also determine when no answer is supported by the paragraph and abstain from answering. SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions but also abstain when presented with a question that cannot be answered based on the provided paragraph[30].

3.3.2 Parameters of Different DNNs

In Table 3.2, we present some of the important parameters of five deep neural network models. These parameters describe some of the aspects of practical implementation of these deep neural networks.

Table 3.2: Comparison of the State of the Art Models

Parameter	BERT	XLNet	RoBERTa	DistilBERT	ALBERT
Model Date	Oct 11, 2018	May 21, 2019	May 26, 2019	Oct 2, 2019	Sep 26, 2019
Method	Bidirectional Transformer, MLM & NSP	Bidirectional Transformer with Permutation based Modeling	BERT Without NSP using Dynamic Masking	BERT Distillation	BERT with Reduced Parameters & SOP
Layers/ Hidden/ Attention Heads	Base: 12/768/12 Large: 24/1024/16	Base: 24/1024/16	Base: 12/768/12 Large: 24/1024/16	Base: 6/768/12	Base: 12/768/12 Large: 24/1024/16
Parameters (in Millions)	Base:110 Large:340	Base: 110 Large: 340	Base:125 Large:355	Base:66	Base:12 Large:18
Training Data	BookCorpus+ English Wikipedia =16GB	Base: 16GB BERT Large: 16GB BERT+ 97GB Additional =113GB	16GB BERT+ CCNews + OpenWebText +Stories =160GB	BookCorpus+ English Wikipedia =16GB	BookCorpus+ English Wikipedia =16GB
Training Time	Base:8 x V100 x 12d Large:280xV100 x1d	512TPU x 2.5days (5 times more than BERT)	1024 x V100 x 1d (4-5 times higher than BERT)	8 x V100 x 3.5d	Base: - Large: 1.7 Times faster
Performance	Outperform all models of Oct 2018	2-15 % Higher Performance over BERT	88.5 on GLUE	97% of BERT base's Performance on GLUE	89.4 on GLUE

3.3.3 Comparison of Performance With SQUAD dataset

In the following table3.3, we present the performance of those five deep neural network models in terms of EM score and F1 score.

Table 3.3: Performance on SQUAD2.0

Method	Date	Performance	
		EM	F1 Score
BERT (Single model) by Google AI Language	Nov 09, 2018	80.005	83.061
XLNet (Single model) by Google Brain & CMU	Nov 15, 2019	87.926	90.689
RoBERTa (Single model) by Facebook AI	Jul 20, 2019	86.820	89.795
DistilBERT[24] (Single model) [a]	-	66.259	69.670
ALBERT (Single model) by Google Research & TTIC	Sep 16, 2019	88.107	90.902
Human Performance by Rajpurkar & Jia et al. Stanford University	2018	86.831	89.452

[a]This result is from[24].

Except for the results of DistilBERT, all others were from the SQUAD2.0 test dataset. These results were obtained from the SQUAD2.0 leaderboard. The result of DistilBERT is from [24] and this is the scores were originally obtained with SQUAD2.0 DEV dataset.

If we carefully study the Table 3.2 and Table 3.3, even all models outperform human performance and also we can understand that the outstanding performance of the ALBERT model. Because the models XLNet, RoBERTa have high parameters in nature, its fine-tuned on SQuAD2.0 performance is low. On the other hand, ALBERT with less number of parameters in nature and the fine-tuned ALBERT on SQuAD2.0 outperform other models with human performance. And also ALBERT was able to do it in a very fast way with less computational resources. From those tables, we can understand that the optimization of models by understanding problems may build a lite and super-efficient model in future.

For better understanding, we present the following bar chart (Fig. 3.10) for clear visualization of the differences in EM Score of the compared models.

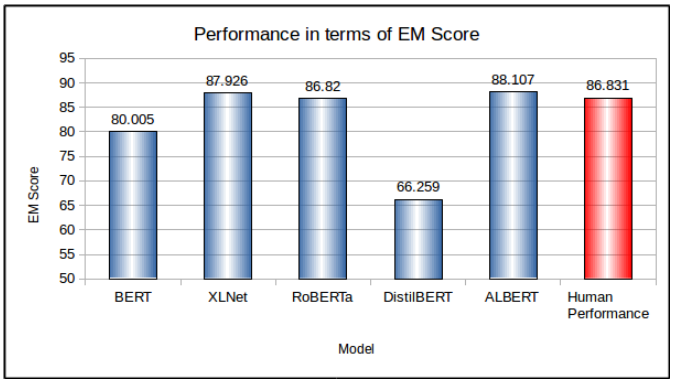

Figure 3.10: Performance of the Models in terms of EM-Score

For better understanding, we present another bar chart (Fig. 3.11) for clearly visualization of the differences in the F1 score of the compared models.

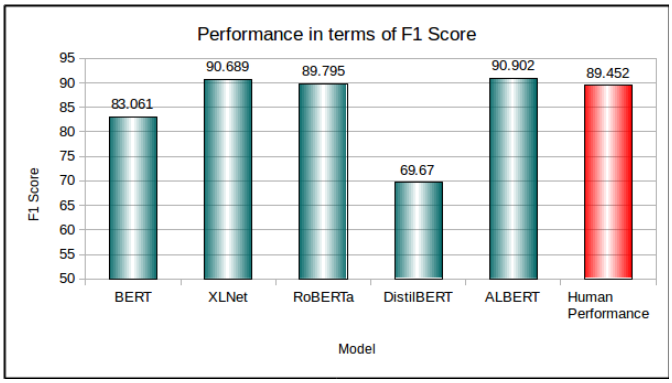

Figure 3.11: Performance of the Models in terms of F1-Score

3.4 Summary

In this work, we did a study on five transformer-based BERT models by analyzing, visualizing, and comparing them with one another.

CHAPTER 4

CHATBOT SYSTEM DESIGN

4.1 Modules of the Project

- Chat Server Module
- Domain-specific QA
- Open-domain QA
- Chatbot Personality QA
- Chat Client Module
- HTML Part
- CSS Part
- Java Script Part

4.2 Hardware Specification

- System: Desktop/Laptop
- Processor: Intel Core i7 2.4 GHz.
- Hard Disk: 500GB SSD.
- Monitor Display: 14’
- Mouse: Optical Mouse.
- Ram: 8Gb.
- Keyboard: 101 Keyboard.

4.3 Cloud Environment

- Processor: Intel(R) Xeon(R) CPU @ 2.20GHz
- Internal Memory Capability: 13Gb
- Disk Capability: 32Gb

4.4 Software Specification

- Front-end: Python, html, CSS, Javascript
- Tools/Libraries: Flask, ngrok, Wiki & DuckDuckGo API, Summarizer
- Operating System: Linux based system
- IDE : Google’s Colabatory with Jupyter environment

CHAPTER 5

PROPOSED CHATBOT ARCHITECTURE

5.1 The Proposed System Architecture

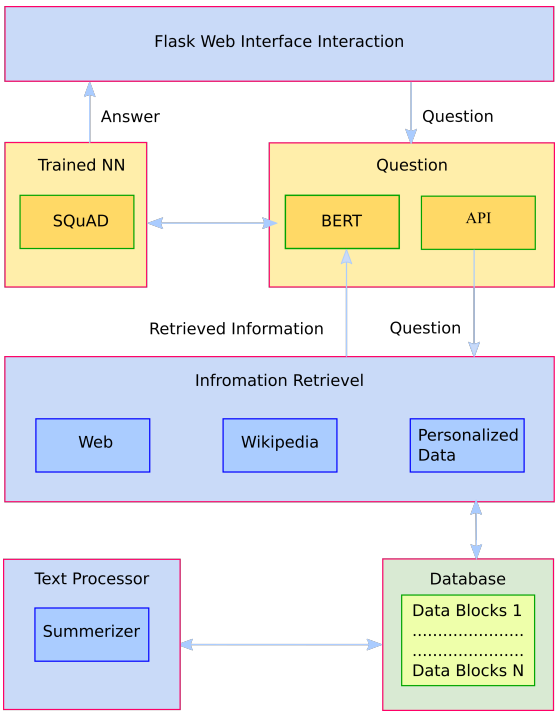

Figure 5.1: Proposed System Architecture.

The “Fig. 5.1” describes the abstract design of the proposed system. It can be divided into five blocks as follows:

- Flask Web Interface Interaction - It is the client-side module. This helps the user(client) to raise the question to the proposed chatbot system.

- Question - The question block contains two sub-blocks. The “API” sub-block caries the question to the next block by processing some logical conditions with a question. Then, the “BERT” sub-block was bidirectionally-connected with two external blocks. The function of the “BERT” sub-block will be explained in consecutive blocks.

- Information Retrieval - This block is responsible for retrieving the information. It retrieves the information with the help of three sub-blocks namely: 1)web retrieval, 2)Wikipedia articles retrieval, and 3)personalized data retrieval. The first two sub-blocks use the internet for information retrieval, so it is Open-domain in nature. But, the specialty of the third sub-block is retrieving information from any customized data(may include database/with or without internet). So it can perform like a functional, personalized chatbot for specific-domain and it was adaptive in nature.

- Database - The database mentioned here may be an offline or online-based QA database. In the case of Open-domain, Wikipedia and DuckDuckGo data were fetched by using two API’s as mentioned in the previous block. In the case of Domain-specific, the offline database mode will be used to retrieve data from any personalized dataset.

- Text Processor - It is a simple summarizer. It is used to summarize large text information from database blocks into meaningful summary

text.

- Trained NN - This block contains a fine-tuned language model(DNN), that is present inside sub-block “BERT” of the “Question” block. To visualize the process, the “BERT” sub-block was placed inside the “Question” block. And, this “BERT” sub-block was responsible for retrieving information from the “Information Retrieval” block and fed it into the trained NN (actually BERT/SQuAD). Then, the trained NN produces the predicted answer to the (user)client by sending it to the “Flask Web Interface Interaction” block. From the above blocks, we can understand the life cycle of a QAS session.

5.2 The Training Process

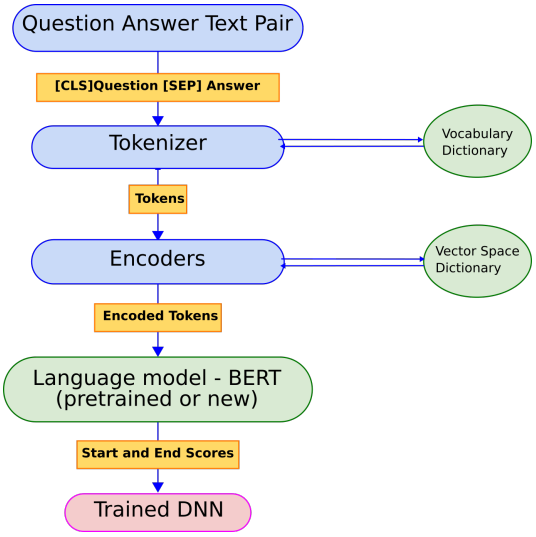

Figure 5.2: The Training Process.

The “Fig. 5.2” , represents the step by step process of training of an language model(DNN). In the first step, it was clear to understand that the question-answer text pair was fed into the language model. Then, the intermediate step next to the first step describes the feeding process, that the

process of using two functions: classifier and separator. Using these functions, the question text was followed by a classifier, and the answer text was separated from the question with the help of a separator, and the end of the answer text was also separated by a separator. Then the question-answer text pair fed into the third step. In this step, it describes the tokenization process, that the process of converting a sequence of word data(input) into independent tokens, this conversion process was carried by a dictionary called a vocabulary dictionary. Then these tokens are fed to the next consecutive steps. The fifth step of the training process, describes the encoders, that the process of encoding(converting) tokens(tokenized words with vocabulary representation) into encoded tokens(machine understandable language), this encoding is done with the help of a dictionary called vector space dictionary. The vector space dictionary is nothing but numerical values, which hold the information of magnitude and direction, and it was also helpful to measure the distance between dependent and independent values. Generally, tokenizer to encoders steps are jointly called as word embeddings. Then, these encoded tokens are taken into the next consecutive steps. This step was called as the Language model(DNN), a DNN specially innovated for Q&A tasks. So the training approach was different from other DNNs. Because most of these language models use attention mechanisms, it is the unsupervised process of extracting 'key - values' using attention over input sequences. The 'key - values' are the relationship between keys and values, and the key was developed from queries, every key has a specific value that is the vectors. The vectors help to evaluate the distance from one key to another key using distance measurement algorithms(like euclidean distance). Generally, the language model performs a few complex mathematical functions, while pre-training a Q&A dataset.

5.3 The Testing Process

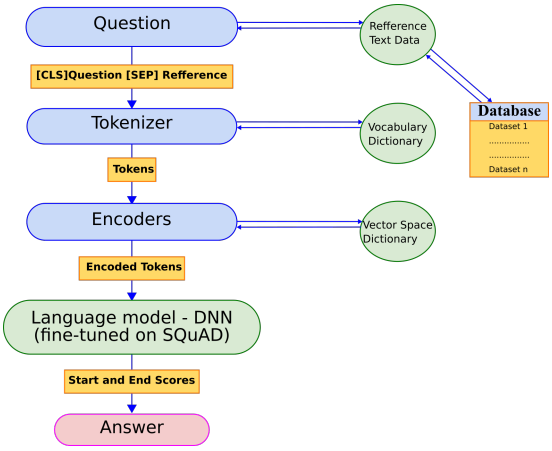

Figure 5.3: The Testing Process.

The “Fig. 5.3” , explains the step by step process of testing a language model(DNN). In the first step, it is clear to understand that the question was taken into a database, which holds the reference text data. Then, the question with reference text was fed into the intermediate next step. In this intermediate step, it describes the feeding process, that a process of using two functions: classifier and separator. Using these functions, the question text was followed by a classifier, and reference text was separated from the question with the help of a separator, and the end of the reference text was also separated by a separator. Then the question reference text pair feed into the next consecutive steps. The next consecutive steps are the same steps seen in the “Fig. 5.2”. Only the major difference is in the fourth step as seen in the “Fig. 5.3”, which is a trained language model and it is used to provide the answer text for the respective question text from the particular reference text as this can be seen as the result in the final step.

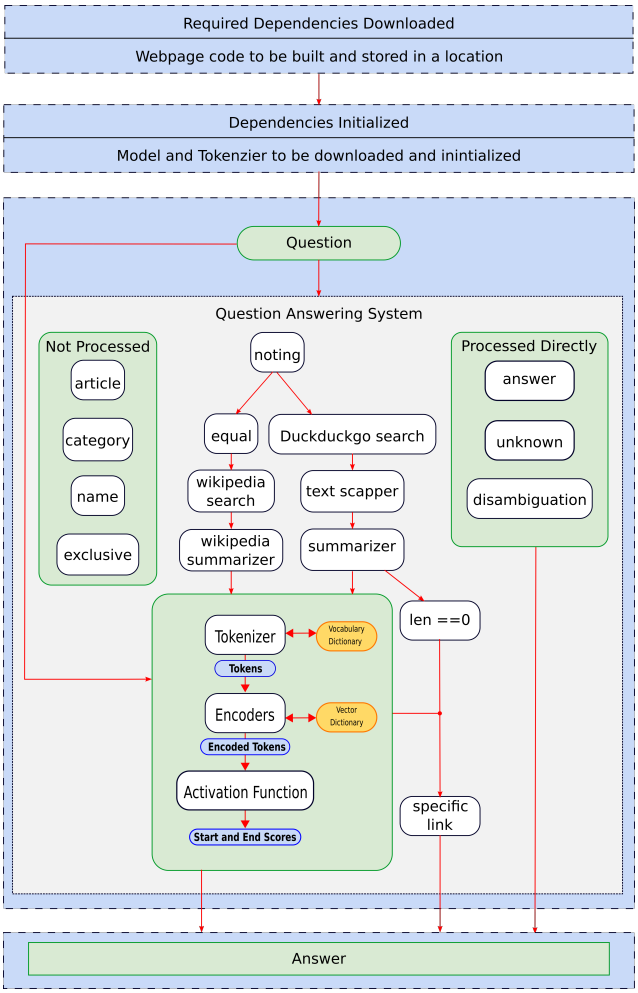

Figure 5.4: The Working Logic of Overall ChatBot.

5.4 The Working Logic of the Overall ChatBot

The “Fig. 5.4”, explains the complete working logic of Open domain mode of the chatbot and that is also divided into the client and server modules of the chatbot. As this system “Fig. 5.4” was implemented in the cloud environment and its working logic diagram was different from traditional implementation. So it is divided into four blocks namely,

- Block One: This block describes the important server module dependencies and client module scripts. There are some important server dependencies used for demonstrating this chatbot. The corresponding server dependencies are downloaded from PyPI website using pip

commands. And then the client module scripts are initialized and then stored in the specific directory of the cloud environment.

- Block Two: This block describes the initializing process of the downloaded server dependencies and trained language model(DNN). The imported dependencies are initialized first and then fine-tuned language model with its tokenizer was also downloaded and initialized.

- Block Three: This block is the most important block of the chatbot. Because it describes the complete working logic of the Open domain mode. In the case of Domain specific mode, only the QAS mode will be changed, and there will be no changes in other blocks of the chatbot.

- Block Four: This block describes the answer provided from the "Block Three" and displays the result to the client module of the system.

5.5 Working Logic of Open domain Server QAS module

The "Fig. 5.4", explains the working logic of the Open domain QA server module of the Chatbot. In this mode, the QAS is based on Wikipedia and DuckDuckGo search engines. In the first step, the question is passed into the QAS. And the QAS process by classifying it into three phases:

- Not Processed: This sub-block describes the "not processed elements" of the Open domain QAS module, which are the articles, category, name, and exclusive. If the question comes under this block it does not provide the answer.

- Nothing: In this sub-block, “Noting” decides whether the question is taken to DuckDuckGo or Wikipedia. It is a step by step process.In the case of Wikipedia, first of all, the question will be identified by the Wikipedia search API, If the corresponding information is present, it will be fed into Wikipedia API built-in summarizer, and then the summarized text feed into the trained DNN - language model(as discussed in the above testing process). and provides the answer as output. If Wikipedia can not process the data DuckDuckGo API will perform a search to the corresponding question and provides multiple links as results. Then the top-ranked link is taken into text scraper(Beautiful Soup) to retrieve the text data. Then the retrieved text data get summarize with the help of a summarizer. Finally, the checker step checks for two conditions 1. If the length of the summarized text is equal to zero then it provides a specific link answer as output. 2. If the length of the summarized text is equal to a whole number, then the summarized text feed into the trained DNN - language model(as discussed in the above testing process). and provides the answer as output. If the question comes under this block it takes a few seconds to provide the answer.

- Processed directly: In this sub-block, the elements “answer”, “unknown”, and “disambiguation” are directly processed by the DuckDuckGo module. Then provide the answer as output. If the question comes under this block it takes a few milliseconds to provide the answer.

5.6 Working Logic of Domain-specific Server QAS module

The “Fig. 5.5”, explains the working logic of the Domain-specific QA server module of the Chatbot. It was divided into five sections namely, 1)Jokes, 2)Personality 3)Personalized Database, 4)Text processor and 5) Language model.

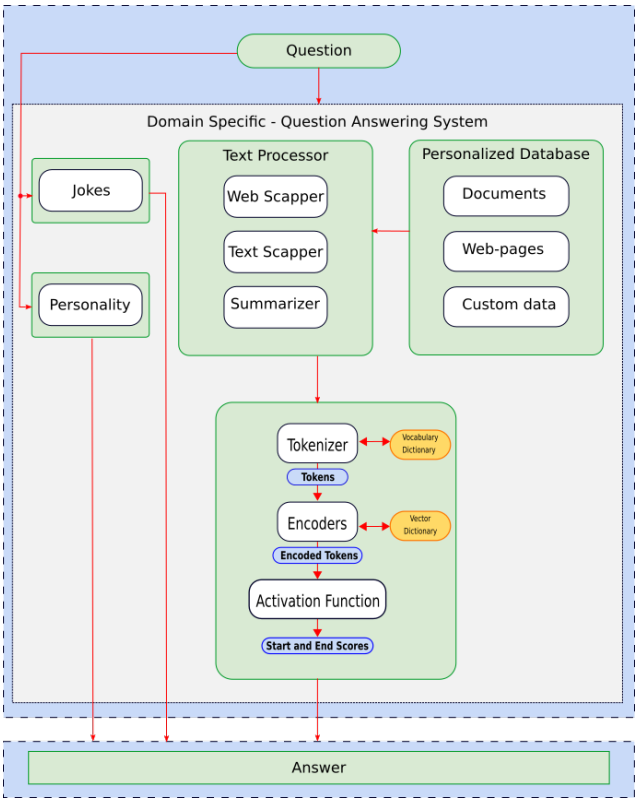

Figure 5.5: The Working Logic of Domain-specific QAS module

- Jokes - It is a computer science based joke module called “PyJokes” implemented in python on the server module, As like modern realtime chatbot systems, it can also tell a joke to the user and entertain him/her. If the client’s question is “tell me a joke”, then it comes under this block and provide joke as the answer.

- Personality - It is a custom dataset prepared by chatbot developers. It

is about the chatbot's bio-data and its interests. This dataset makes the chatbot behave like a human. If the client's question is "tell me about you? or who are you?", then it comes under this block and provides its personal information as the answer.

- Personalized database - It was a dataset prepared by the annotators/developers/information scrappers or prepared automatically from facts, documents, webpages, FAQ, books, etc. It can be customized and personalized according to the client's needs. This makes this dataset as "Domain-specific in nature.

- Text Processor - It was a text processing unit mainly focus on creating human-understandable text from scarped data(web, documents, or custom dataset). Then summarizing those texts whit the help of a summarizer. This summarizer is used to make large text sequences into a meaningful text summary.

- Language model - It was the identical block that was already described in "Fig. 5.3". It is used to process Domain specific questions with the Domain specific reference data. The question with reference data is processed into the fine-tuned language model and provides its predicted result as an answer. If the client's question is "where is TCE or when TCE was established?", then it comes under this block and provides its specific information as the answer.

5.7 Experimental Setup

As discussed in the above sections that this project requires a lot of computing resources, modules were developed in such a way. The chatbot server-

side module is made run on the google cloud environment. Even the client-side module will be served from the server to the client machine and then made run at the client's browser.

5.8 Summary

This Chapter explains about the proposed ChatBot architecture, In which the question is processed to produce the relevant answer. The second section and third explains about the training and testing process, that occurs in a language model(DNN). Then, the forth and fifth section describes the working logic of the Open-domain QAS, Then, the sixth section describes the working logic of Domain-specific QAS. Finally, the last section describes about the experiential setup of the proposed system.

CHAPTER 6

IMPLEMENTATION, TESTING, RESULTS AND DISCUSSION

6.1 About the Text Dataset sources

The SQuAD, Wikipedia articles, Web Search based on DuckDuckGo and personalized dataset are the text datasets used in this paper.

6.1.1 Stanford Question Answering Dataset (SQuAD)

SQuAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable[18]. SQuAD2.0 combines the 100,000 questions in SQuAD1.1[19] with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable

ones. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph[18].

In Proposed system, the deep neural networks already trained with SQuAD are used as base neural network for performing our Question Answering System.

6.1.2 Wikipedia

Wikipedia is a digital encyclopedia with millions of information based on multi-domain. In the Proposed system, the Wikipedia API for python is used for information retrieval from Wikipedia articles.

6.1.3 DuckDuckGo

DuckDuckGo is an internet search engine that maintains defending searcher's privacy and avoiding the filter bubble of personalized search results. It distinguishes itself from other search engines by not profiling its users and by showing all users the same search results for a given search term. In the Proposed System, the DuckDuckGo API for python is used for information retrieval from DuckDuckGo searched web pages.

6.1.4 Personalized Dataset

This dataset is a custom dataset prepared by experts, it may be structured or unstructured text data. This dataset is prepared by the process called scrap-

ing. These personalized data are scrapped from FAQs, Documents, Webpages of an institute or organization with legal permission and confirmation by the specific institute or organization. In the Proposed System, the Scrapping tools for python are used for preparing Personalized Dataset.

6.2 Testing

This is a chatbot system that concentrates on both Open-domain and Domain-specific and made as an open-source project. The testing was an important part of this system. Generally, Python follows PEP 8 standard for styling and analysing the codes. So, we had tested the structure of our system using Pylint toolkit.

6.2.1 Structure Testing using Pylint

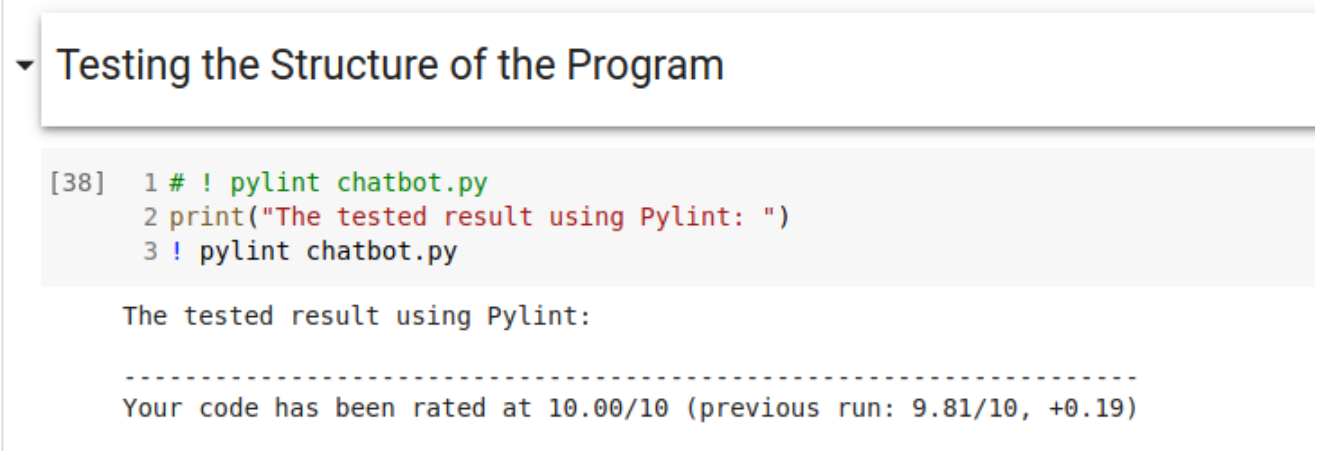

Figure 6.1: Chatbot Structure testing using Pylint.

As mentioned earlier, this project was implemented in the cloud environment. The code is saved using the write command and tested inside the respective code block. The Fig. 6.1, shows the Pylint style and logic score of the chatbot's main code. While running this testing, some of the exceptions were intentionally disabled. Because, it refers to the parent pretrained

models and libraries, which cannot be modified. This project only provides a design concept of an AI chatbot.

6.3 Observations and Findings

If we carefully study the five phases of NLP presented in our survey on NLP in chapter 2, it is obvious that almost all the algorithms or models invented up to the first four phases of NLP history were only achieved 'Below Human level Performance'. Only during the fifth phase of NLP history the algorithms and models achieved 'Near Human level Performance', 'Equal Human level Performance', and even 'Above Human level Performance'. And DNNs played a important role in achieving this levels of performance.

6.3.1 BERT based chatbot

In chapter 2, we described various transformer-based models and we tried BERT fine-tuned on SQuAD as our first DNN model that to be implemented in our chatbot. But it was a complete disaster. Even though these transformer-based language models were good in the SQuAD leader board, the main problem was with document accessing and retrieving, which are only capable of processing the small documents or processing the documents as chunks, and sometimes it may result in out-of-memory problems. So we understood that BERT based models that we had surveyed were not helpful for our problem. So we came with another language model called Longformer.

6.3.2 Longformer based chatbot

Longformers was a transformer-based model[15], that is capable of processing long text documents. As our project is meant for making an AI chat-

bot, it often deals with long documents. So to avoid chunking or to reduce out-of-memory problems. Our chatbot was implemented using the Longfomer language model. Hopefully, It was already fine-tuned with SQuAD2.0. So It can be used for building our AI chatbot. By using this Longfromer with Open-domain mode APIs and Domain-specific dataset, this project server module is built. And providing client interface, the server module is connected with client module with the help of flask framework with ngrok server.

6.3.3 Results of the Chatbot

The Results are provided in the below sections, that based on chat conversations by the user to the chatbot. The chatbot is made up of two modes, Open-domain and Domain-specific. If the question is based on an Open-domain it would use the APIs: DuckDuckGo and Wikipedia search engines. The AI chatbot has some logical reason to understand the query(question), by understating the nature of the query, it will decide to use Wikipedia or DuckDuckGo search engine. Then, the query was parsed into the following APIs and fed into the language model. Finally, the answer is displayed to the client. In the case of Domain-specific AI chatbot mode can be changed for a particular dataset and process the same procedure and displays the answer to the client.

6.3.4 Results from PC

The Fig. 6.2 shows the screenshot of the web application running on the client screen. In that application, the client asks two questions to the chabot, which was set on Open-domain mode - web. Both questions were based on the Coronavirus. It was clear that the chatbot was able to fetch appropriate information from the web by using DuckDuckGo and Wikipedia APIs and

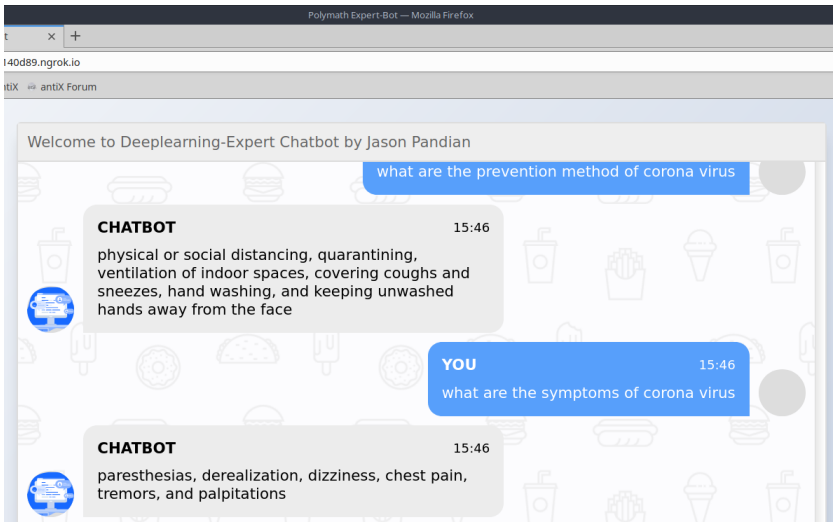

Figure 6.2: Open Domain mode of the Chatbot

provide appropriate results with the help of the fine-tuned language model.

6.3.5 Result of Domain-specific mode

The Fig. 6.3 shows the screenshot of the web application running on the client's mobile, In that application, the client asks the questions about Tamilnadu college of engineering(TCE) to the chabot. The chatbo was set on Domain-specific mode. In Domain-specific mode, the chatbot replies accurately to all the questions asked by the client. So we can understand that this system can be implemented for any domain easily and perform well without any extra training or fine-tuning.

6.4 Comparison of results with search engines

The Fig. 6.4 shows the screenshots of the web application running on the client computer. In this, the client asks a question about Mahatma Gandhi to the Google search engine, DuckDuckGo search engine, and our chabot. The DuckDuckGo search engine only provides a summary of Gandhi. But Google provided a specific answer. Like Googe, our chatbot also capable of

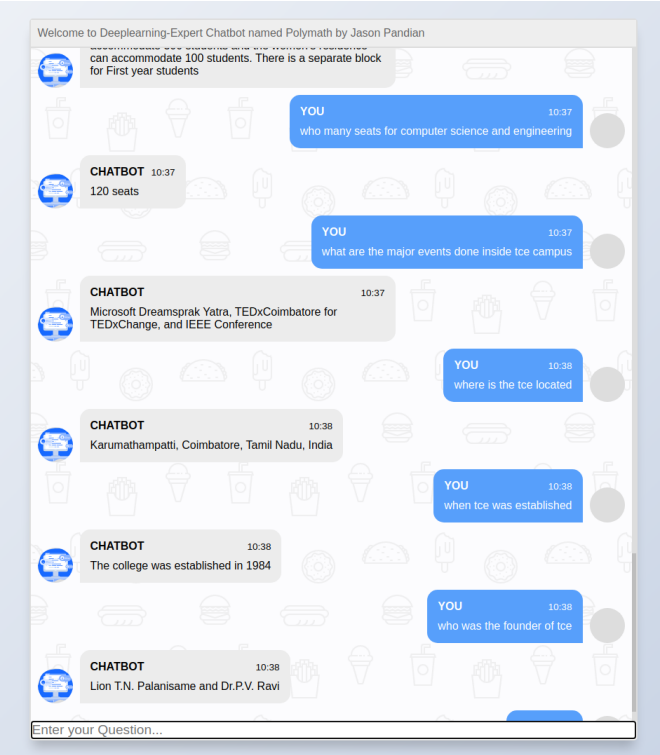

Figure 6.3: Domain-specific mode based on TCE

providing specific answer.

6.5 Result that Outperforms Google Assistant

The Fig. 6.5 describes the screenshot of the web application running on a client's mobile, In this, the client asks a question about a Tamil Television Serial to the Google Assistant and our chabot. But our Chatbot outperforms Google Assistant by providing an accurate answer. Because nowadays there are unusual pop-up ads and paid advertisements to come first in popular search engines/chatbots. It because of its business-oriented recommendation system in the name of enhancing user experience by collecting user's privacy. Using open-source deep neural networks and privacy concerns API like DuckDuckgo and few packages helps to make a clean chatbot that will provide a clear, specific, and purposeful result.

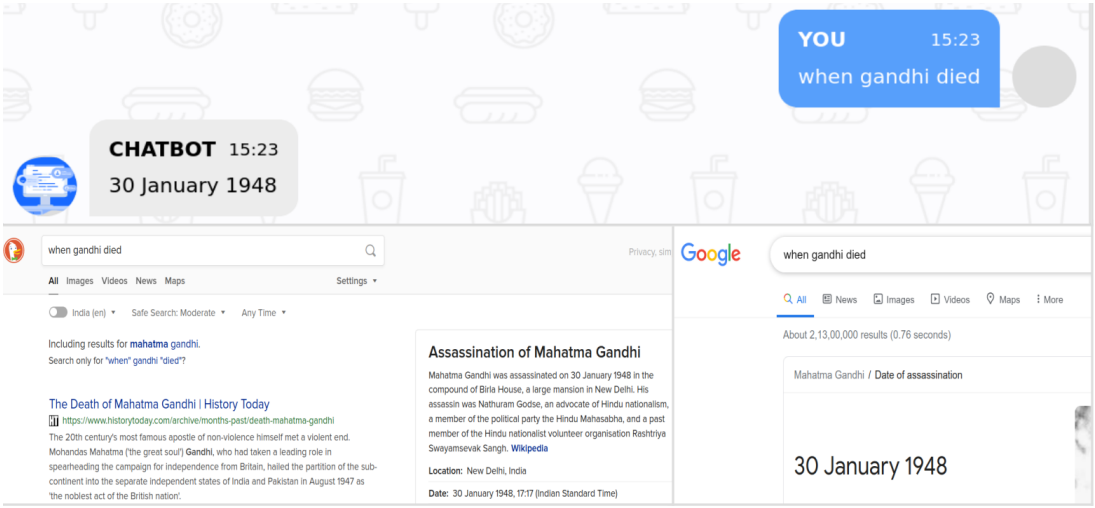

Figure 6.4: Comparision of Google, DuckDuckGo and Chatbot

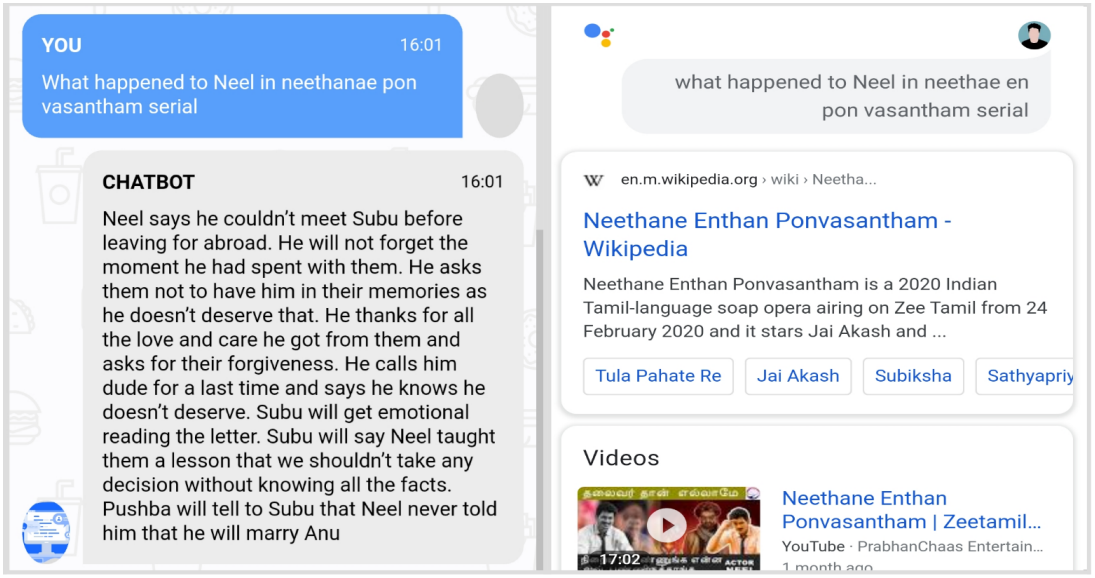

Figure 6.5: Outperforms Google Assistant

6.6 Summary

We have successfully implemented a AI Chatbot using Deep Learning based NLP techniqus. The implemented system performed very well. From the results of the previous section, it is clear that the chatbot was able to respond with appropriate information.

CHAPTER 7

CONCLUSION AND FUTURE WORK

We have successfully implemented an AI Chatbot using Deep Learning based NLP techniques. The implemented system performed very well. From the results of the previous section, it is clear that the Chatbot was able to respond with appropriate information.

As we observed in our survey, DL based NLP models performed poorly even until 2017 and their results are very lower than human performance. After that, with the availability of heavily powered machines, DL based NLP models started to perform good but their performance was still a little bit low comparing with actual human performance. During 2018, some State of the Art models evolved and played a huge role in NLP tasks and achieved equal to human performance. After that, Multi powered(layered or ensembled) State of the Art models evolved and started to achieve above human level performance in various NLP tasks. As we mentioned in the previous section, the NLP models based on Deep Learning are quite interesting and

were able to achieve above human level of performance.

Today, DNN based NLP systems capable to translate any language text or speech to any other known language. In the future, NLP may become super-intelligent and there may be a possibility of fine-tuned NLP system that may decipher the ancient, unknown, language scriptures and hence replace the traditional language expert. Future NLP model may provide human-like or better human-like Bots/Chatbots that can be your future caretaker, companion, or next-generation virtual teacher. The future NLP model may show a possibility to communicate with aliens(if there will be something) that often fantasize in best-selling science-fiction novels and fiction emulated movies. By surviving past and present of NLP and DNN, this survey ends with an endnote, that one day humans may achieve DNN/NLP/AI systems that can be super-intelligent and can be optimized to solve present unsolved problems of Quantum Physics and other Mathematical secrets of nature.

In this work we used only one language(English) for training and testing our chatbot. But every DNN based NLP system is Language independent models. So that we can train a chatbot to any regional language. Even it is possible to train the same bot with more than one language. One may address the issues in developing a multi-lingual chatbot system.

APPENDIX A

MORE RESULTS

A.1 PersonalityDataset

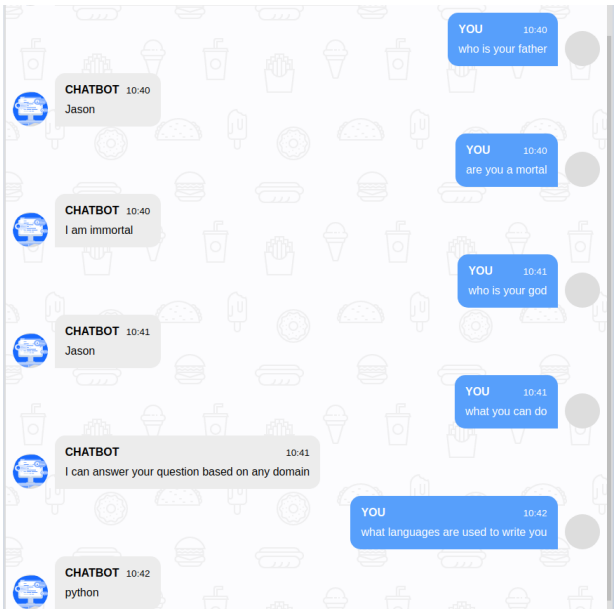

Figure A.1: Chatbot Personality.

The Fig. A.1 shows the client having a personal conversation with the chatbot.

A.2 Chatbot that Can Tell a Joke

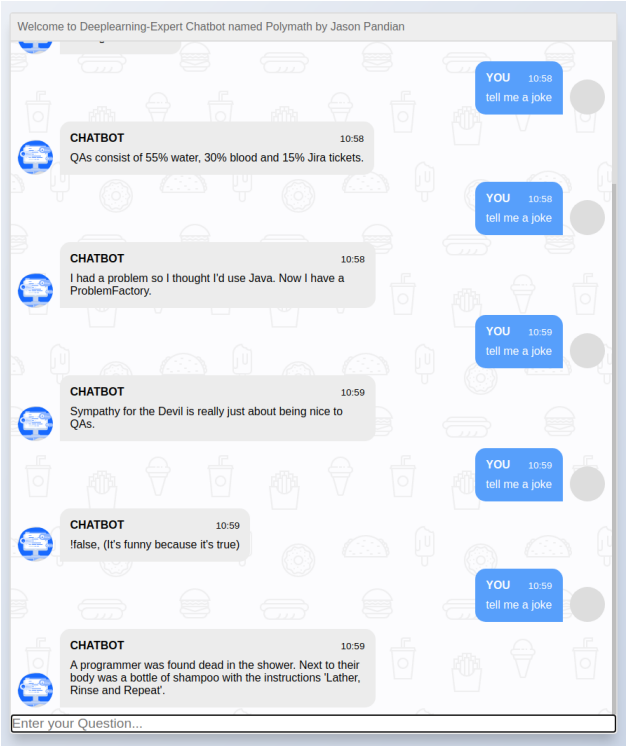

Figure A.2: Chatbot that can Joke

The Fig. A.2 shows a funny conversation with the chatbot. It can be programmed to make jokes related with Programming.

A.3 Indirect Question Answering

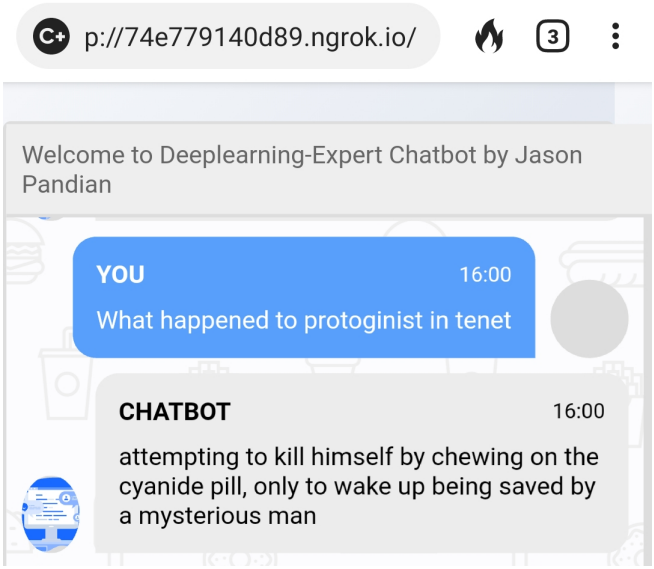

Figure A.3: Open Domain Chatbot Result of Indirect QA.

The Fig. A.3 shows the screenshot of the the web application running on the client mobile. Using this application the client asks indirect question to the chatbot about a English movie 'Tenet'. The chatbot tries to understand the question and give accurate results.

A.4 Answering to Random Questions

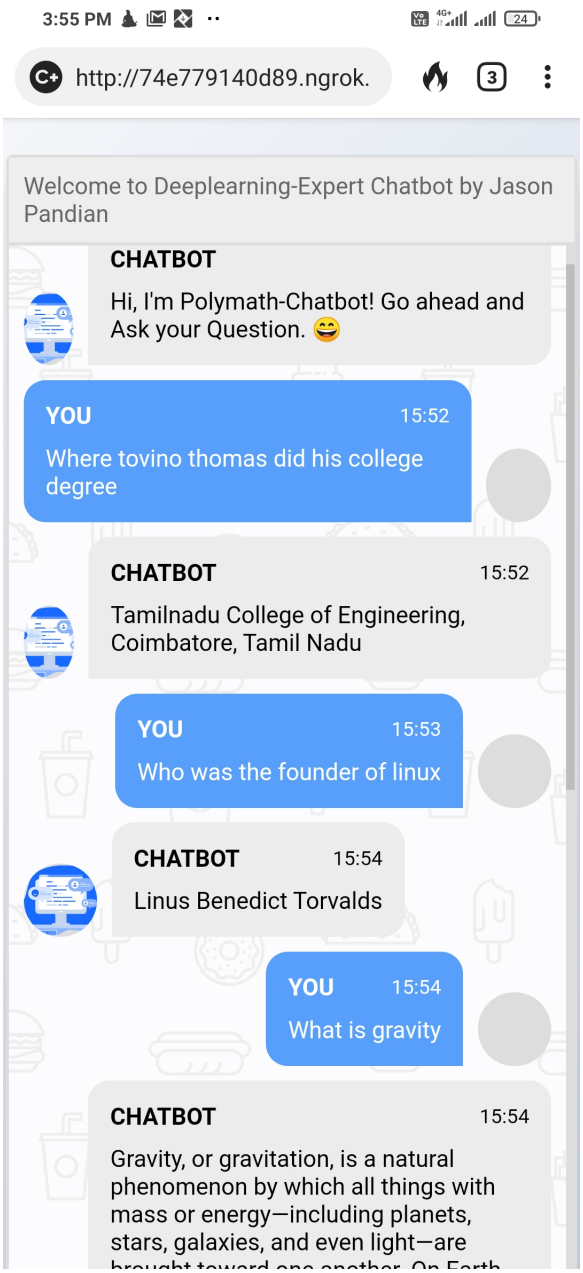

Figure A.4: Random Question with Accurate Answers

The Fig. A.4 shows the screenshot of the the web application running on the client mobile. Using this application the client asks a question to the chatbot. The chatbot tries to understand the question and give accurate answer.

APPENDIX B

CODE SAMPLES

B.1 Main ChatBot Server Module in Python

```
#import files
print("Modules Setting up")
from flask import Flask, render_template, request
from flask_ngrok import run_with_ngrok
import numpy as np
import torch
import duckduckgo
import wikipedia
import requests
from bs4 import BeautifulSoup
from gensim.summarization import summarize
from transformers import LongformerTokenizer,
    LongformerForQuestionAnswering
import pyjokes
print("Modules Setup Completed Sucessfully")

print("DNN Setting up")

```

```
18 tokenizer = LongformerTokenizer.from_pretrained("valhalla/longformer-base
       -4096-finetuned-squadv1")
19 model = LongformerForQuestionAnswering.from_pretrained("valhalla/
       longformer-base-4096-finetuned-squadv1")
20 print("DNN Setup Completed Sucessfully")
21 global context
22 context = 'web'
23
24 global tce_info
25 tce_info = '''Tamil Nadu College of Engineering also known as TCE is
       situated at Karumathampatti,
26 ........
27 ....... '''
28
29
30 global self_info
31 self_info = '''Hi I am Polymath - Chatbot with multi personalities,
       ............
32 ........
33 ........ '''
34
35
36 # %%writefile chatbot.py
37 def chatbot():
38     """
39     main code of the chatbot.
40     """
41     global model, tokenizer, app, context, tce_info, self_info
42     try:
43         app = Flask(__name__)
44         run_with_ngrok(app)
45
46         # starts ngrok when the app is run
47         @app.route("/")
48         def index():
49             """
50             open wi and dispay html.
51             """
52             return render_template("index.html")
```

```
53
54          @app.route("/get")
55          def get_bot_response():
56              """
57              bot response
58              """
59              # get the user question
60              question = request.args.get('msg')
61              bert_needed = False
62              response = duckduckgo.query(question)
63              print('Response Type:', response.type)
64              # print(response.json)
65              global context
66
67              if 'context' in question and 'web' in question:
68                  context = 'web'
69                  return str('Setting Context web')
70
71              if 'context' in question and 'tce' in question:
72                  context = 'tce'
73                  return str('Setting Context tce')
74
75              if 'you' in question or 'your' in question:
76                  encoding = tokenizer.encode_plus(question, self_info,
                        return_tensors="pt")
77                  input_ids = encoding["input_ids"]
78                  attention_mask = encoding["attention_mask"]
79                  start_scores, end_scores = model(input_ids,
                        attention_mask=attention_mask)
80                  all_tokens = tokenizer.convert_ids_to_tokens(input_ids
                        [0].tolist())
81                  answer_tokens = all_tokens[torch.argmax(start_scores):
                        torch.argmax(end_scores) + 1]
82                  answer0 = tokenizer.decode(tokenizer.
                        convert_tokens_to_ids(answer_tokens))
83                  return str(answer0)
84
85              if context == 'tce':
86                  encoding = tokenizer.encode_plus(question, tce_info,
```

```
                return_tensors="pt")
87              input_ids = encoding["input_ids"]
88              attention_mask = encoding["attention_mask"]
89              start_scores, end_scores = model(input_ids,
                    attention_mask=attention_mask)
90              all_tokens = tokenizer.convert_ids_to_tokens(input_ids
                    [0].tolist())
91              answer_tokens = all_tokens[torch.argmax(start_scores):
                    torch.argmax(end_scores) + 1]
92              answer1 = tokenizer.decode(tokenizer.
                    convert_tokens_to_ids(answer_tokens))
93              return str(answer1)
94          if context == 'web':
95
96              if question == 'tell me a joke':
97                  joke = pyjokes.get_joke(language='en', category='
                        neutral')
98                  return str(joke)
99              if response.type == 'article':
100                 print('Not Yet Procecced')
101             elif response.type == 'disambiguation':
102                 answer_text = str(response.related[0].text)
103                 print(answer_text)
104             elif response.type == 'category':
105                 print('Not Yet Procecced')
106             elif response.type == 'name':
107                 print('Not Yet Procecced')
108             elif response.type == 'exclusive':
109                 # print(response.json)
110                 # answer_text=response.answer.text
111                 print("Sorry! I am a Chatbot. Not a Calculator.")
112                 return str("Sorry! I am a Chatbot. Not a Calculator."
                        )
113             elif response.type == 'nothing':
114                 # print(r.json)
115                 bert_needed = True
116                 url = str(duckduckgo.get_zci(question))
117                 print(url)
118                 # url = 'ddldzvdzvn'
```

```
119                     link = url
120                     if link == url:
121                         try:
122                             wiki = url.rsplit('/', 1)[-1]
123                             print(wiki)
124                             answer_text = wikipedia.summary(wiki,"lxml")
125                             print("Wikipedia",answer_text)
126                             print("Len of Wikipedia", len(answer_text))
127                         except:
128                             page = requests.get(url).text
129                             print("DuckDuckgo")
130                             # Turn page into BeautifulSoup object to
                                    access HTML tags
131                             soup = BeautifulSoup(page)
132                             # Get text from all <p> tags.
133                             p_tags = soup.find_all('p')
134                             # Get the text from each of the ""p tags and
                                    strip surrounding whitespace.
135                             p_tags_text = [tag.get_text().strip() for tag
                                     in p_tags]
136                             # Filter out sentences that contain newline
                                    characters
137                             # '\n' or don't contain periods.
138                             sentence_list = [sentence for sentence in
                                    p_tags_text if not '\n' in sentence]
139                             sentence_list = [sentence for sentence in
                                    sentence_list if '.' in sentence]
140                             # Combine list items into string.
141                             article = ' '.join(sentence_list)
142                             answer_text = summarize(article, word_count
                                    =500)
143                             print("Wikipedia",answer_text)
144                             print("Len of Wikipedia", len(answer_text))
145                             # ratio=0.3
146                             if (len(answer_text) == 0):
147                                 bert_needed = False
148                                 answer_text = "Sorry, But You can  Check
                                        out this Link: " + url
149                                 # print(answer_text)
```

```
150                 elif response.type == 'answer':
151                     answer_text = duckduckgo.get_zci(question)
152                     print(answer_text)
153                 else:
154                     print('Response Type: Unknown')
155                     answer_text = str(duckduckgo.get_zci(question))
156                 if bert_needed == True:
157                     # return (str('Thinking.Please Wait'))
158                     encoding = tokenizer.encode_plus(question,
                            answer_text, return_tensors="pt")
159                     input_ids = encoding["input_ids"]
160                     attention_mask = encoding["attention_mask"]
161                     start_scores, end_scores = model(input_ids,
                            attention_mask=attention_mask)
162                     all_tokens = tokenizer.convert_ids_to_tokens(
                            input_ids[0].tolist())
163                     answer_tokens = all_tokens[torch.argmax(start_scores)
                            :torch.argmax(end_scores) + 1]
164                     answer = tokenizer.decode(tokenizer.
                            convert_tokens_to_ids(answer_tokens))
165                     if answer == '':
166                         return (str("Sorry, But You can  Check out this
                                Link: " + url))
167                     return str(answer)
168                 else:
169                     return str(answer_text)
170
171     except NameError:
172         print()
173
174     try:
175         if __name__ == "__main__":
176             app.run()
177     except NameError:
178         print()
179
180
181 chatbot()
```

B.2 Main HTML Client Interface

```
<!—%%writefile /content/templates/index.html-->
<!DOCTYPE html>
<html>
<head>
    <title>Polymath Expert-Bot</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
        jquery.min.js"></script>

<style>
 ..........
 ..........
<!—%% CSS code removed to minimize space in listing-->
 ..........
 ..........
</style>
</head>
<body>
<section class="msger">
  <header class="msger-header">
    <div class="msger-header-title">
      <i class="fas fa-comment-alt"></i> Welcome to Deeplearning-Expert
          Chatbot named Polymath by Jason Pandian
    </div>
    <div class="msger-header-options">
      <span><i class="fas fa-cog"></i></span>
    </div>
  </header>

  <main class="msger-chat">
    <div class="msg left-msg">
      <div
       class="msg-img"
       style="background-image: url(https://www.messagely.com/wp-content/
           uploads/2019/09/Chat-bots-hero-img.svg)"
      ></div>

```

```
35        <div class="msg-bubble">
36          <div class="msg-info">
37            <div class="msg-info-name">CHATBOT</div>
38
39          </div>
40
41          <div class="msg-text">
42            Go ahead and Ask me anything. Type 'context tce' for domain
                  specific or 'context web' to get back to web mode.
43          </div>
44        </div>
45      </div>
46    </main>
47                  <div id="userInput">
48                      <input id="textInput"  class="msger-input" type="
                            text" name="msg" placeholder="Enter your Question
                            ..."  autofocus="autofocus" />
49                  </div>
50    </section>
51    <script>
52      const msgerForm = get(".msger-inputarea");
53      const msgerInput = get(".msger-input");
54      const msgerChat = get(".msger-chat");
55
56
57      // Icons made by Freepik from www.flaticon.com
58      const BOT_IMG = "https://www.messagely.com/wp-content/uploads
            /2019/09/Chat-bots-hero-img.svg";
59      const PERSON_IMG = "https://pixabay.com/get/5
            fe7d6474c52b114f6df8278c2283e7c083edbec55517941712d79.svg";
60      const BOT_NAME = "CHATBOT";
61      const PERSON_NAME = "YOU";
62      function appendMessage(name, img, side, text) {
63        //   Simple solution for small apps
64        const msgHTML = `
65          <div class="msg ${side}-msg">
66            <div class="msg-img" style="background-image: url(${img})"></
                  div>
67
```

```
68            <div class="msg-bubble">
69              <div class="msg-info">
70                <div class="msg-info-name">${name}</div>
71                <div class="msg-info-time">${formatDate(new Date())}</div>
72              </div>
73
74              <div class="msg-text">${text}</div>
75            </div>
76          </div>
77        msgerChat.insertAdjacentHTML("beforeend", msgHTML);
78        msgerChat.scrollTop += 500;
79      }
80    // Utils
81    function get(selector, root = document) {
82      return root.querySelector(selector);
83    }
84
85    function formatDate(date) {
86      const h = "0" + date.getHours();
87      const m = "0" + date.getMinutes();
88
89      return `${h.slice(-2)}:${m.slice(-2)}`;
90    }
91
92    function random(min, max) {
93      return Math.floor(Math.random() * (max - min) + min);
94    }
95
96    function getBotResponse() {
97
98        var rawText =  $("#textInput").val();
99      $("#textInput").val("");
100       $.get("/get", { msg: rawText }).done(function (data) {
101       appendMessage(BOT_NAME, BOT_IMG, "left", data);
102       });
103   }
104   $("#textInput").keypress(function (e) {
105       if (e.which == 13) {
106         var rawText = $("#textInput").val();
```

```
107             // const msgText = msgerInput.value;
108             //if (!msgText) return;
109             appendMessage(PERSON_NAME, PERSON_IMG, "right", rawText);
110             getBotResponse();
111         }
112     });
113   </script>
114 </body>
115 </html>
```

REFERENCES

[1] Alex Graves, Greg Wayne, Ivo Danihelka, “Neural Turing Machines”, Google DeepMind, arXiv:1410.5401v2 [cs.NE], 2014.

[2] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, and others, “Hybrid computing using a neural network”, 2016.

[3] Amit Mishra, Sanjay Kumar Jain, “A survey on question answering systems with classification”, Journal of King Saud University - Computer and Information Sciences (2016) 28, 345-361.

[4] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, Richard Socher, “Ask Me Anything: Dynamic Memory Networks for Natural Language Processing”, CA USA, arXiv:1506.07285v5 [cs.CL], 2016.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”, Google Brain, arXiv:1706.03762v5 [cs.CL] 6 Dec 2017.

[6] Collobert, Weston, “Machine Learning, Proceedings of the Twenty-Fifth International Conference”, Univ.Helsinki, Finland, 2008.

[7] David H. Hubel and Torsten N. Wiesel, “Brain and visual perception: the story of a 25-year collaboration”, Oxford University Press US. p. 106. ISBN 978-0-19-517618-6, 2005.

[8] Do-Hyoung Park, Vihan Lakshman,“Question Answering on the SQuAD Dataset”, Stanford University, 2018.

[9] Dzmitry Bahdanau, Cho Kyunghyun, Yoshua Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate”, ArXiv:1409.0473, 2014.

[10] Finlay, Janet, and Alan Dix, “An introduction to Artificial Intelligence”, UCL Press, London, 1996

[11] Fukushima, K., “Neocognitron”, doi:10.4249/scholarpedia.1717, 2007.

[12] Fukushima, Kunihiko, “Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”, Biological Cybernetics, 1980.

[13] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, “Sequence to sequence learning with neural networks”, ArXiv:1409.3215, 2014.

[14] Ivakhnenko, A. G., Lapa, V. G., “Cybernetics and Forecasting Techniques”, American Elsevier Publishing Co. ISBN 978-0-444-00020-0, 1967.

[15] Iz Beltagy, Matthew E. Peters and Arman Cohan , ”Longformer: The Long-Document Transformer”, Seattle, WA, USA, arXiv:2004.05150v2 [cs.CL] 2 Dec 2020.

[16] Jacob Devlin, Ming-Wei, Chang Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, Google AI Language, arXiv:1810.04805v2 [cs.CL] 24 May 2019.

[17] Jason Pandian, “Difference Between Traditional Neural Network and Deep Neural Network”, A Tech Blog, https://jason.co.in/difference-between-traditional-neural-network-and-deep-neural-network/

[18] Jason Pandian, “Understanding Convolutional Neural Network”, A Tech Blog, https://jason.co.in/understanding-convolutional-neural-network/

[19] Jason Pandian, “Understanding Recurrent Neural Network”, A Tech Blog, https://jason.co.in/understanding-recurrent-neural-network/

[20] Jason Weston, Sumit Chopra, Antoine Bordes, “Memory Networks”, Facebook AI Research, arXiv:1410.3916v11 [cs.AI], 2015.

[21] Karen Sparck Jones, “Natural Language Processing: A Historical Review”, Computational Linguistics, vol. 9-10; Pisa, Dordrecht, 1994.

[22] Keith D. Foote, “A Brief History of Deep Learning”, 2017. https://www.dataversity.net/brief-history-deep-learning/

[23] Linnainmaa, Seppo, “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”, Univ. Helsinki, 1970

[24] Melanie Beck, “Evaluating QA: Metrics, Predictions, and the Null Response”, Cloudera Fast Forward Labs, Minneapolis, MN, Jul, 2020.

[25] Mohammad Nuruzzaman and Omar Khadeer Hussain , ”A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks”, IEEE 15th International Conference on e-Business Engineering (ICEBE).

[26] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang, “SQuAD: 100,000+ Questions for Machine Comprehension of Text”, Stanford University, 2016.

[27] Rosenblatt F, “The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain”, Psychological Review Vol. 65, No. 6,1958.

[28] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, “End-To-End Memory Networks”, New York University, arXiv:1503.08895v5 [cs.NE], 2015.

[29] SquAD(closed), https://rajpurkar.github.io/SQuAD-explorer/explore/1.1

[30] SquAD2.0, https://rajpurkar.github.io/SQuAD-explorer/

[31] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, “Efficient Estimation of Word Representations in Vector Space”. ArXiv:1301.3781, 2013.

[32] Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”, arXiv:1910.01108v4 [cs.CL] 1 Mar 2020.

[33] Williams, Ronald J.; Hinton, Geoffrey E., Rumelhart, David E. , “Learning representations by back-propagating errors”. Nature. 323 (6088): 533-536, ISSN 1476-4687. October 1986.

[34] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, Paul G., “RoBERTa: A Robustly Optimized BERT Pretraining Approach”, arXiv:1907.11692v1 [cs.CL], 26 Jul 2019.

[35] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut, “ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS”, A conference paper at ICLR 2020, arXiv:1909.11942v6 [cs.CL] 9 Feb 2020.

[36] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le,“XLNet: Generalized Autoregressive Pretraining for Language Understanding” , Carnegie Mellon University, Google AI Brain Team, rXiv:1906.08237v2 [cs.CL] 2 Jan 2020.