Opinion forming Interactive AI based on NLP/NLU, focused on Economics

By

Rutwik Rajanagouda Patil

Student ID: 2270387

Supervisor: Dr Mohammed Bahja

A thesis submitted to the University of Birmingham

For the degree of MSc in Artificial Intelligence and Machine Learning

School of Computer Science

University of Birmingham

Birmingham, UK

September 2022

Table of Contents

List of Figures

Abstract

Intelligent Conversational AI's that mimics human which is hard to build. It has been on-going research to build a conversational AI that can have its own opinion. It is also another challenge to build one which infers its knowledge from a simple database. Firstly, this project tries to solve dataset problem, which is on contrast, as seen in the previous works which require large and complex datasets, reflecting in heavy weight model training. Thus, using simple JSON format dataset any researcher from other field can modify the dataset to build relevant chatbot specializing in that area. Secondly this project proposes a recipe to build such an intelligent AI which can form its opinion, through several experimentations. This work proves that traditional Natural Language Processing methods such as Bag of Words, Stemming, Word2Vec fail in creating an advanced AI, so does having a standalone model for entire chatbot. The research done proves that using new state-of-the-art techniques of Transformers and BERT embeddings, combined with multi-model neural network approach works the best. Thus, creating a successful conversational AI that can mimic humans providing opinions on the topic of Economy. Further using golden standard of evaluation, it proves to be quite successful, having around 80% success rate. Finally, this paper addresses the future work and direction to propel further research.

Keywords

Chatbot, Text Summarization, Sentiment Analysis, Transformers, Conversational AI, Blended Skill Talk(BST)

Acknowledgement

I would like to express my sincere appreciation and gratitude to my supervisor Dr. Mohammad Bahja, this research would not have been completed without his mentorship and guidance. I would also like to thank my parents(Dr. Rajanagouda G Patil and Dr. Nayan R Patil) whose support has been pivotal in encouraging, supporting, and inspiring me throughout my life.

A special mention to my grandfather(Mr. Chinagouda B Chinagudi) who was the first engineer in our family and played the key role in planting the seeds of curiosity and problem-solving in me from a very young age. My parents and my grandfather are my main source of motivation for my continued research and dedication to building projects that impact and help the society.

I would also like to thank my project inspector Prof. Hamid Dehghani for his inputs, constructive feedback, and support during this research. His teachings of experimentation and research enabled me to embark on a journey to complete this research project.

Chapter I: OVERVIEW

1.1 Introduction

Chat bots which are also known as conversational bots are entire software solutions that simulate conversations with human beings in a natural language. Today, in the real world there are many conversational agents which we use. For example, Apple Siri, Microsoft Cortana, Amazon Alexa, Google Assistant, ecommerce shopping chat bots and web-based chat bots. A lot of progress has been made in designing chatbots, right from rule-based chat bots to chat bots that incorporate machine learning and almost feel human like.

However, building chatbots with opinion forming capabilities and Blended Skill Talk is often very challenging. Recent progress has been made in this area but still the process of opinion forming is largely unexplored, and this research is done to develop chatbots with such capability.

1.2 Motivation

This research aims to implement a chatbot with the capability of generating its own opinions on the topic of user interest based on its own knowledge dataset. Also, present it in the form of a conversation that engages the user with its opinions and views. This project is propelled by the zest to create a chatbot that contemplates the below mentioned ideas and research:

- Building a chatbot that can form opinions on the given subject matter from its knowledge database and can also have conversations is extremely useful in understanding AI opinion formation.
- These intelligent chatbots can also enhance Human Computer Interaction (HCI), making the interactions with the chatbot unique and relevant to the chosen subject matter. This also expands the user accessibility for the wider community, thereby reducing the effort and resolution time needed to explore the topic of Economics.
- The AI is to have a deep understanding of the concepts and topic of the knowledge dataset i.e.., about Economics. This is essential in enhancing the conversation, adding to the fact that it enables us to have human like conversations and get the AI's opinion based on its facts.

1.3 Research Objectives

The scope of applications of this project is vast depending upon the usage. Hence careful research is an integral part of it. Through exploration of the present generation Natural Language Processing and Natural Language Understanding like Transformers and well-established techniques like Bag of Words, Vectorisation and sentence similarity need to be integrated supporting each other on different levels to obtain the desired output. Also, there is a significant amount of biological influence in form of human brain study and human decision making. We will list the following research considerations:

- The main research objective was to build an opinion forming chatbot from a small amount of data i.e., on Economics, a useful case in topic specific chatbots. Also, a different dataset format for the research has been created than one used in previous methods [14, 21, 23, 25, 26, 28, 29], to enable users/ computer hobbyists to make their own customisation of AI knowledge base.
- It is a known fact that the human brain is an amazing machine, and we are learning new things about it every day. The research objective was also to loosely mimic the human conversation as our brain interpret it based on knowledge/information retrieval, conversation logic and finally opinion forming and present it as conversation.
- For our dataset or similar datasets, the previous methods [14, 21, 23, 25, 26, 28, 29] fail. The main reason being that dataset is specific, structure simplified and hence can be limited in size. Thus, the approach of single standalone model fails. Upon further research the multi-modal approach using combination of NLP techniques along with state-of-the-art transformer works the best and this is evident in the experiments performed.

1.4 Methodology

This research project is divided into several parts, which have been titled under different experiments. The experiments, as evidently shown in Chapter IV, will progress sequentially through the demonstration of basic NLP techniques (Bag of Words, Vectorisation, Word embeddings), dataset creation via web scrapping, text summarisation, sentiment analysis(using Transforms Neural Networks), Chatbot interface design (on Python) and finally see how these combined techniques together contribute towards reaching our goal of creating an opinion forming AI chatbot.

This chapter expresses the motivation for the undertaken research and why it is so important to keep the research on-going in this direction. This is a wider question that has been a hot ground for research and exploration among Computer Linguistics.

Before moving on to the next chapter lets just quickly go over about main algorithm and see the process flow that our chatbot uses when interacting with the humans. Firstly, we get the input in the form of user query or question via chat bot interface, then we turn the input string into a BERT vector space encoding. In the next step we calculate the similarity between user query and question patterns from the database and get most likely match (cosine similarity score of 60% or higher). Then we retrieve the corresponding knowledge from the database by our first pretrained Neural Network. Later if the extracted text is too long (above 50 words) we perform summarisation of the retrieved text via second Neural Network (Transformer 1). Then we perform sentiment analysis using the third Neural Network (Transformer 2) on the generated text to generate the possible opinion(Positive or Negative). Finally, we use the technique of Blended skill talk to put together the results obtained from the neural networks in the form of a conversation string through the chatbot application interface in a humanly presentable way.

In the next chapter we will go through the previous works done, where we draw inspiration, appreciate the progress made and utilise the research to further our investigation. We will explore research papers on conversational AI’s Chatbot recipes, Transformer architectures for various Natural Language Processing/Natural Language Understanding tasks and see their relevance to our research.

Then finally in Chapter V we will see the conclusions drawn from our experiments and how we compare our results. Also, we will see what the limitations are our Artificial Intelligence based opinion forming Chatbot has. Then will also set the course for the journey for further research my exploring possible ways to improve this.

Chapter II: LITERATURE REVIEW

Introduction

In this section we will have a look at the previous work done by computer scientists around the world in the field of Natural Language Processing, Natural Language Understanding and specifically in developing methods to create opinion generation and chatbots systems. This will help us appreciate the research progress and give us a sense of direction to further work on.

2.1 Literature Review

Opinion generation is hot topic in the field of NLU for AI. Building an AI which can plan and choose how to respond to the environment in the form of NLU is a challenge, scholars around the world are working on. These decisions making techniques are based in both real-world data and on the options of the programmers and engineers. However, what’s interesting is talking about the intellectual capacity to feel, to understand life and human world.

Some notable work is done in this field, and they are, [22] which discusses on giving generating prior opinion using the Neural Network and then train it by collecting opinions from 5 different users and updated its internal neural network to match the opinions. Authors propose to use an attention-based abstract generation model — a data-driven approach trained to generate informative, concise, and fluent opinion summaries. Next work is [21], authors present an attention-based neural network model that can absorb information from multiple text units to construct informative, concise, and fluent summaries. An importance-based sampling method is designed to allow the encoder to integrate information from an important subset of input. This system is also rated as more informative and grammatical in human evaluation according to the paper. The work [23] is by far one of the most advanced and is based on multiple transformer models which addresses problems of dialogue, discussion. This paper examines several diverse aspects of intelligence and summarises the key results. The model known as Gopher lifts the performance over current state-of-the-art language models across roughly 81% of tasks containing comparable results, notably in knowledge-intensive domains such as fact checking and general knowledge. Next work [25] defines a generative model for a review collection which capitalizes on the intuition, when generating a new review given a set of other reviews of a product with control of uniqueness. Results show that the model is capable to produce fluent and coherent summaries reflecting common opinions. In the paper [27] authors propose a sentence ordering system. Training is done for feature extraction for main sentence to be the claim and next sentences are ordered based on a ranking by the system. The next work [28] an attention-based network is proposed to absorb information from multiple text units to construct informative, concise, and fluent summaries using importance-based sampling designed to allow the encoder to integrate information. Authors in paper [26] purpose an attention-based encoder and decoder system for opinion question generation problem for interacting with customers in online shopping platforms. The model can identify key parts of the sentences and generate questions that are non-redundant relevant to the user. The paper [29] presents a unique approach to opinion-based text generation that non-text data such as image and metadata also contains extra information, hence a multi modal system opinion summarization framework is created. The framework obtains a representation of each modality using a separate encoder for each modality, and the text decoder generates a summary. First an encoder is created for text only then a separate encoder for image, then finally a decoder for text output. Framework training is done end-to-end by combining all modules. Thus, there is still scope for opinion-based systems to improve and build upon specific domains especially generating opinionated conversational sentences.

Chapter III: BACKGROUND

3.1 Natural Language Processing

Let’s get to know what Natural Language Processing is. Right now, as we are reading and writing this report, we our brain is actually forming some sort of comprehension from it. When we program a computer do it, it is known as Natural Language Processing. Natural Language Processing is a very useful in all sorts of textual based AI applications. NLP starts with unstructured text, which is how we speak and interpret text. For example, some unstructured text is "add cheese and bread to my grocery list." Now we understand exactly what that means, but it is unstructured to a computer. So, what we proceed to do, is to create a structured representation of that same information that a computer can process. Now list of groceries can be considered as a big list and then it has elements within it, like cheese, and bread. Now this is a structured list. The job of natural language processing is to translate between unstructured and structured data sitting right in the middle.[ref]

Natural Language Processing is used in various Machine Learning and Artificial Intelligence applications to derive insights from textual data. Corporations, scientists, and governments use it across the world to understand user behaviour, sentiments, opinion and much more. Let’s go through the tasks that can be accomplished by Natural Language Processing. These include language translation, similar sentence identification, paragraph summarization, text classification (e.g., spam message filtering), sentiment analysis (e.g., Positive, Neutral or Negative user feeling), question answering, conversational agents and chatbots. This paper uses many of the above-mentioned tasks and we will go through them in details in the upcoming sections.

3.2 Natural Language Understanding

Let’s recall the structured representation of language example we used above i.e., “add cheese and bread to my grocery list”. Now let’s ask the question, how can we teach the computer to understand the comprehension of this statement? This is where Natural Language Understanding come into play. NLU is often regarded as a hard problem in Artificial Intelligence, until recently when time series and attention based neural networks were introduced. Traditional Machine Learning problems weren’t enough to capture the meaning of the sentence and also understand how position of each word and played a key role in understanding text better. The tasks of NLU extend from NLP but differ in performance as it has context understanding capabilities.

3.3 Conversation AI and Chatbots

Conversation AI or Conversational agents are software’s that programmed to have conversation with users. They can be either simple rule-based agents, more complex like retrieving relevant data and forming meaningful conversations or even more sophisticated ones that can have human like conversations on a wide range of topics. However, it is necessary that a large amount of database knowledge in the form of conversations or access to internet is need for many of these conversational agents to get the desired output.

Chatbots are also a form of conversational agents except the input given and the response obtained is in the form of text only in an interface(command line or standalone application).

Regardless of the presentation both follow that almost similar processing architecture internally. It is very important to set the objectives for the AI agent and build the software accordingly. Some of the main points to consider and follow upon when designing these conversational agents would be as follows:

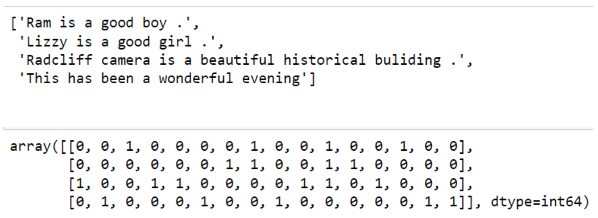
1. Scope: This refers to the domain or topic on which the conversation agent can converse on. In the view of this project, it is Economics. Also, this research stresses on the opinion generation on user query, a hot topic for Computer Linguists these days. These objectives help us focus on the outline, goals and objectives to prepare the methodology accordingly.
2. Responses: Conversational agents are expected to have human like conversations. People today have a limited time to get as accurate information as possible. Thus, the agent has to generate shorter responses which make sense and are simplified.
3. Human element: This is an interesting, exciting capability programmed into a chatbot, but at the same time very hard to implement. Some easy parts include giving agent a name, or a voice. It is fact that users will want to interact with agents who sound and interact human-like. This is all about the conversation engagement.

3.4 Bag of Words

As we have seen earlier computers can't understand language like we do. Computers understand numbers, more specifically into vectors. Bag of words is one of the ways to achieve this. So basically, bag of words is way to represent some text mathematically for modelling a machine learning algorithm [1]. It's based on the concept that if two sentences have matching words then they are similar. Bag of words is modelled after 2 things.

1. Generate a vocabulary of all the words and characters in the entire corpus of given sentences.
2. Measure which words are present in which sentences by creating a vector.

Bag of words is usually combined with techniques of Lemmatisation or stemming after removing stop words from the sentences to get final vector representation. The downside of this technique is that it does not take into regard into order or structure which could reveal more meaningful information about the sentences. This is evident in the experiments done during this project.



```
['Ram is a good boy .',
 'Lizzy is a good girl .',
 'Radcliff camera is a beautiful historical buliding .',
 'This has been a wonderful evening']
```

```
array([[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0],
       [1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1]], dtype=int64)
```

Figure 1 Bag of Words representation with Code

3.5 Stemming

Words in any language originate from a main word, these are referred as Stem words and derivative words branch out of this stem. These branched words mean the same but have different forms depending upon the grammar of the sentences. Now we want these branched words to be reduced to their corresponding stem words. Stemming in Natural Language Processing does exactly this, reduces words to their stem words. For example, let's consider the word '*stem*' itself. This can take forms such as '*stemming*' or '*stemmed*'. So, the words '*stemming*' or '*stemmed*' have to be reduced to '*stem*'. This is very beneficial and efficient to create a shorter word corpus for the given sentences.

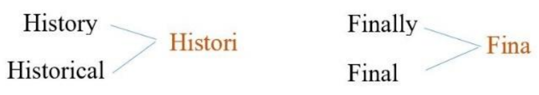
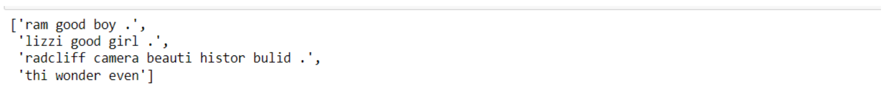


Figure 2 Showing stemming of word History & Finally [2]

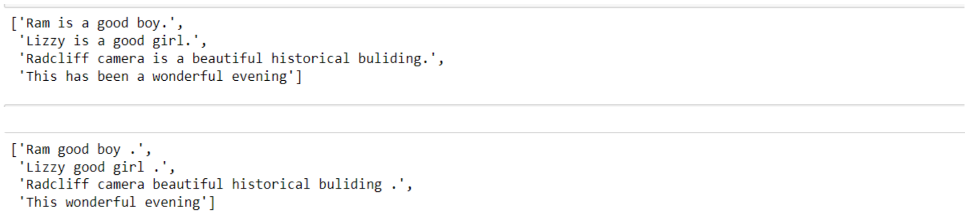


```
['ram good boy .',
 'lizzi good girl .',
 'radcliff camera beauti histor bulid .',
 'thi wonder even']
```

Figure 3 Stemming implementation with output

3.6 Lemmatization

Lemmatization is very similar to stemming. It also reduces the words to their root stem, but there is a catch. Stemming usually reduces words to a form where they lose their meaning. Lemmatization overcomes this by keeping this to the minimum. However, this technique sometimes can lead to some same stem words not to be detected efficiently. But it works well compared to stemming from a word understanding point of view. This is evident from the above figure. The word '*Histori*' does not ring a bell instantly. After lemmatisation the word '*historical*' is kept as it is. This helps in creating a more meaningful corpus, thus is very useful, since the parts of speech are preserved. Lemmatization's limitations are overcome by word to vector embedding. This is explained in corresponding section.



```
['Ram is a good boy.',
 'Lizzy is a good girl.',
 'Radcliff camera is a beautiful historical buliding.',
 'This has been a wonderful evening']
```

```
['Ram good boy .',
 'Lizzy good girl .',
 'Radcliff camera beautiful historical buliding .',
 'This wonderful evening']
```

Figure 4 Lemmatization implementation with output

3.7 Cosine Similarity

When we have huge list of vectors and want to find out how similar are new or other vectors, similarity learning comes into place. This is done by simply measuring the inner dot product between the input vector and the target vectors by definition. This means that mathematically it is calculated by the cosine angle between two vectors. This helps in knowing whether these two vectors point towards the same direction and how close they are in vector space. The equation of it is simple and is given below.

$$\cos\theta = \ similarity(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{||\boldsymbol{x}||\,||\boldsymbol{y}||}$$

Figure 5 Formula to calculate cosine similarity

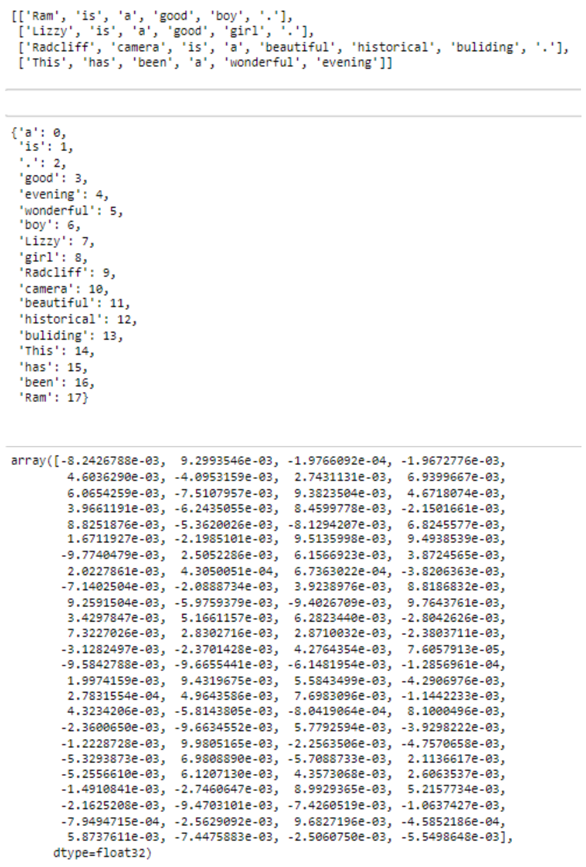
As it turns out, this technique is very handy in Natural Language Processing. Particularly helping to know how similar two sentences are. A very easy and handy method in creating conversational agents. Users can use provide any query sentence and AI has to match with its stored patterns in database to understand what exactly the user is speaking about. This eventually lead us close to Natural Language Understanding and is used in the final version of chatbot created.

3.8 Word to Vector and Word Embedding

As we have seen what Bag of Words, Stemming and Lemmatization do and how they work an important drawback is observed, i.e., they don't preserve semantic information. What this means is that the relationship between the words is not known. Let's take a simple example, suppose we have three words '*man*', '*woman*' and '*play*'. Immediately we human can perceive that '*man*' and '*woman*' are almost similar in the sense both refer to human but different genders that all. Now we need to make computers understand this. This is where the word to vector comes.

In this model each word is represented as vector of multiple dimensions (32 or more). This vector representation is called word embedding and depends on what kind of model we are using (such as Word2vec, GloVe or BERT embeddings). Thus, semantic information is preserved in this way. Also, computers can build a graph internally which can be used to traverse to understanding semantics. Let's outline the steps to create a word to vector representation:

1. Tokenize the sentences to convert them into word tokens
2. Create a histogram of the tokens obtained
3. Then create a matrix of all the unique tokens and represent the occurrence and relationship between the words and nearby words that lead to it by cosine similarity.



```
[['Ram', 'is', 'a', 'good', 'boy', '.'],
 ['Lizzy', 'is', 'a', 'good', 'girl', '.'],
 ['Radcliff', 'camera', 'is', 'a', 'beautiful', 'historical', 'buliding', '.'],
 ['This', 'has', 'been', 'a', 'wonderful', 'evening']]
```

```
{'a': 0,
 'is': 1,
 '.': 2,
 'good': 3,
 'evening': 4,
 'wonderful': 5,
 'boy': 6,
 'Lizzy': 7,
 'girl': 8,
 'Radcliff': 9,
 'camera': 10,
 'beautiful': 11,
 'historical': 12,
 'buliding': 13,
 'This': 14,
 'has': 15,
 'been': 16,
 'Ram': 17}
```

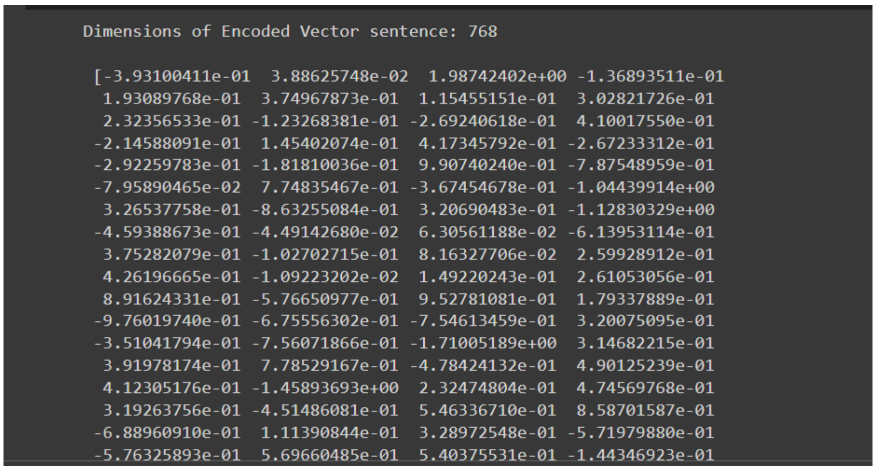
```
array([-8.2426788e-03,  9.2993546e-03, -1.9766092e-04, -1.9672776e-03,
        4.6036290e-03, -4.0953159e-03,  2.7431131e-03,  6.9399667e-03,
        6.0654259e-03, -7.5107957e-03,  9.3823504e-03,  4.6718074e-03,
        3.9661191e-03, -6.2435055e-03,  8.4599778e-03, -2.1501661e-03,
        8.8251876e-03, -5.3620026e-03, -8.1294207e-03,  6.8245577e-03,
        1.6711927e-03, -2.1985101e-03,  9.5135998e-03,  9.4938539e-03,
       -9.7740479e-03,  2.5052286e-03,  6.1566923e-03,  3.8724565e-03,
        2.0227861e-03,  4.3050051e-04,  6.7363022e-04, -3.8206363e-03,
       -7.1402504e-03, -2.0888734e-03,  3.9238976e-03,  8.8186832e-03,
        9.2591504e-03, -5.9759379e-03, -9.4026709e-03,  9.7643761e-03,
        3.4297847e-03,  5.1661157e-03,  6.2823440e-03, -2.8042626e-03,
        7.3227026e-03,  2.8302716e-03,  2.8710032e-03, -2.3803711e-03,
       -3.1282497e-03, -2.3701428e-03,  4.2764354e-03,  7.6057913e-05,
       -9.5842788e-03, -9.6655441e-03, -6.1481954e-03, -1.2856961e-04,
        1.9974159e-03,  9.4319675e-03,  5.5843499e-03, -4.2906976e-03,
        2.7831554e-04,  4.9643586e-03,  7.6983096e-03, -1.1442233e-03,
        4.3234206e-03, -5.8143805e-03, -8.0419064e-04,  8.1000496e-03,
       -2.3600650e-03, -9.6634552e-03,  5.7792594e-03, -3.9298222e-03,
       -1.2228728e-03,  9.9805165e-03, -2.2563506e-03, -4.7570658e-03,
       -5.3293873e-03,  6.9808890e-03, -5.7088733e-03,  2.1136617e-03,
       -5.2556610e-03,  6.1207130e-03,  4.3573068e-03,  2.6063537e-03,
       -1.4910841e-03, -2.7460647e-03,  8.9929365e-03,  5.2157734e-03,
       -2.1625208e-03, -9.4703101e-03, -7.4260519e-03, -1.0637427e-03,
       -7.9494715e-04, -2.5629092e-03,  9.6827196e-03, -4.5852186e-04,
        5.8737611e-03, -7.4475883e-03, -2.5060750e-03, -5.5498648e-03],
      dtype=float32)
```

Figure 6 Word2Vec representation

3.9 Sentence BERT Embeddings

BERT (Bidirectional Encoding Representation from Transforms) is a revolutionary Depp neural network released by Google in 2018. BERT's architecture in a super simplified sense consists of vector encoders stacked one after the other sequentially to create a model that can understand sentences the bidirectional way.

The main problem with the previous methods like Word2Vec was that it used to convert sentences into vector representation successfully, but the semantic interpretation was not great. Each word had a fix representation regardless of its position and mean with the sentence context. An illustration of this would be the following two sentences, '*Rutwik went to the bank to deposit money*' and '*Rutwik enjoyed fishing on the banks of river Thames.*'. Here in these two sentences the word '*bank*' has two different meanings. Word2Vec was not able to differentiate between these and assigned them to the same vector space embedding. However, with Sentence BERT embeddings [4] the vector representation of this is different. This captures the uniqueness of these words and lead to a more accurate feature representation. This makes the difference, moving into Natural Language Processing with a firm step and produces better models.



```
Dimensions of Encoded Vector sentence: 768

 [-3.93100411e-01  3.88625748e-02  1.98742402e+00 -1.36893511e-01
   1.93089768e-01  3.74967873e-01  1.15455151e-01  3.02821726e-01
   2.32356533e-01 -1.23268381e-01 -2.69240618e-01  4.10017550e-01
  -2.14588091e-01  1.45402074e-01  4.17345792e-01 -2.67233312e-01
  -2.92259783e-01 -1.81810036e-01  9.90740240e-01 -7.87548959e-01
  -7.95890465e-02  7.74835467e-01 -3.67454678e-01 -1.04439914e+00
   3.26537758e-01 -8.63255084e-01  3.20690483e-01 -1.12830329e+00
  -4.59388673e-01 -4.49142680e-02  6.30561188e-02 -6.13953114e-01
   3.75282079e-01 -1.02702715e-01  8.16327706e-02  2.59928912e-01
   4.26196665e-01 -1.09223202e-02  1.49220243e-01  2.61053056e-01
   8.91624331e-01 -5.76650977e-01  9.52781081e-01  1.79337889e-01
  -9.76019740e-01 -6.75556302e-01 -7.54613459e-01  3.20075095e-01
  -3.51041794e-01 -7.56071866e-01 -1.71005189e+00  3.14682215e-01
   3.91978174e-01  7.78529167e-01 -4.78424132e-01  4.90125239e-01
   4.12305176e-01 -1.45893693e+00  2.32474804e-01  4.74569768e-01
   3.19263756e-01 -4.51486081e-01  5.46336710e-01  8.58701587e-01
  -6.88960910e-01  1.11390844e-01  3.28972548e-01 -5.71979880e-01
  -5.76325893e-01  5.69660485e-01  5.40375531e-01 -1.44346923e-01
```

Figure 7 SBERT encoding representation from Hugging Face models

3.10 Sentence Similarity via Cosine Similarity and BERT embeddings

As we saw in the above sections, the gradual improvement into Natural Language Understanding, this section deals with putting all these things together. This is done to understand the user query and then run further algorithms to provide the output.

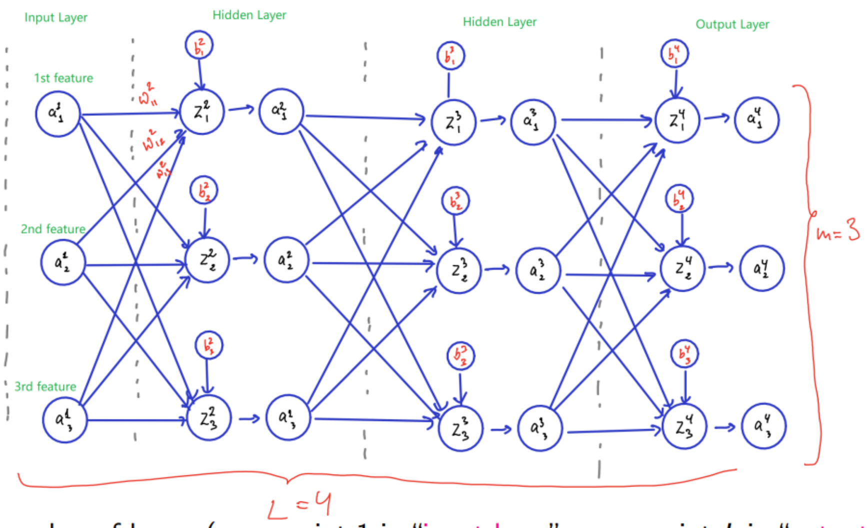
Let's see how we achieve this procedurally:

1. The sentence transformer [4] is used to convert the given user query and patterns from database into sentence BERT transformer encodings by using the model 'sentence-transformers/all-MiniLM-L6-v2' available from Hugging Face library.
2. Then we use the cosine similarity to see which pattern the query it matches. This helps us understand what user means. The threshold similarity score is often set after experimentation.

3.11 Basic Neural Networks

Interpreting and mimicking brain has been a challenge for computer scientists all around the world. The neural networks are the bases of Deep Learning which is a sub field in Machine Learning. This resembles the human brain more particularly Neurons, the building block of our brain hence the name Neural Networks. These networks can recognize and identify relationships between given data and predict the output. These tasks can be regression, classification, time series forecasting across text, numbers, audio, video, and image as inputs.

To achieve these tasks, Neural Networks have many layers in them which in turn have interconnected nodes. As depicted in the figure. These are also called Deep Neural Networks (DNN's), hence the term Deep Learning. The input to these is vector arrays and are fed to input layers. The output layer has several neurons which are 'activated' according to the type of task by defining an activation function. The layers of neurons are called hidden layers. The connection between each neuron is called weight. As the name suggests they hold weight as to how much information/signal they pass.



- L: number of layers (superscript 1 is "input layer", superscript L is "output layer")
- m: "width" of network (can vary between layers)
- ω_{jk}^{ℓ}: "weight" of connection between k-th unit in layer $\ell-1$, to j-th unit in layer l
- b_j^{ℓ}: "bias" of j-th unit in layer l
- $z_j^{\ell} = \sum_k \omega_{jk}^{\ell} a_k^{\ell-1} + b_j^{\ell}$: weighted input to unit j in layer ℓ
- $a_j^{\ell} = \sigma(z_j^{\ell})$: "activation" of unit j in layer ℓ, where σ is an "activation function"

Figure 8 Structure of Deep Neural Network [5]

Before we train our neural network, we need to keep in mind that we cannot find the perfect set of weights for or neural network model. It is hence necessary to find the set of weights that lead to the nearby correct output. Hence, we convert this to an optimisation problem, and this is done mathematically and is quite efficient. Let's define some terms namely Loss function, Forward Propagation, Back propagation, and Gradient descent.

Loss Function: When we train the neural network, we simultaneously check for it output in each step with expected output. The difference between the predicted output and actual output is known as loss and we seek to minimize this loss. The function which defines it is called loss function[6]. Loss function include Mean Squared Error, Cross entropy. Low loss means higher accuracy.

Forward Propagation: It is the method via which the Neural Network makes predictions. The calculation for this is shown in the Figure 9.

Forward Propagation: Example

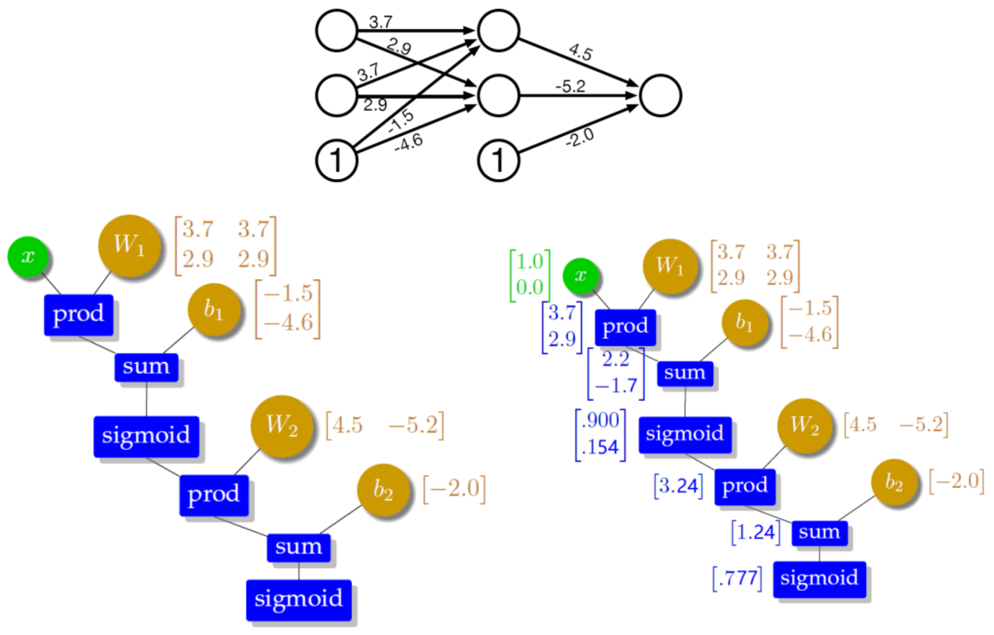
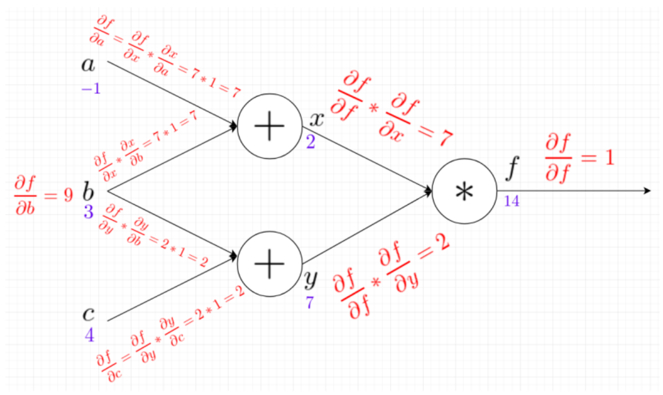

3.7
2.9
3.7
2.9
-1.5
-4.6
1
4.5
-5.2
-2.0
1
x
W_1
$\begin{bmatrix}3.7 & 3.7\\2.9 & 2.9\end{bmatrix}$
prod
b_1
$\begin{bmatrix}-1.5\\-4.6\end{bmatrix}$
sum
sigmoid
W_2
$[4.5 \quad -5.2]$
prod
b_2
$[-2.0]$
sum
sigmoid
$\begin{bmatrix}1.0\\0.0\end{bmatrix}$
$\begin{bmatrix}3.7\\2.9\end{bmatrix}$
$\begin{bmatrix}2.2\\-1.7\end{bmatrix}$
$\begin{bmatrix}.900\\.154\end{bmatrix}$
$[3.24]$
$[1.24]$
$[.777]$

Figure 9 Forward Propagation Calculations[5]

Back Propagation: It is a recursive application of chain rule along a computational graph to compute the gradients of all the weights. It is simple and efficient way to do it for computers., unlike the direct traditional method hence useful for training the Deep Neural Networks.

Another Example: Backward Pass


$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial x} * \frac{\partial x}{\partial a} = 7 * 1 = 7$
a
-1
$\frac{\partial f}{\partial x} * \frac{\partial x}{\partial b} = 7 * 1 = 7$
$\frac{\partial f}{\partial b} = 9$
b
3
$\frac{\partial f}{\partial y} * \frac{\partial y}{\partial b} = 2 * 1 = 2$
c
4
$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial y} * \frac{\partial y}{\partial c} = 2 * 1 = 2$
$+$
x
2
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial x} = 7$
$+$
y
7
$\frac{\partial f}{\partial f} * \frac{\partial f}{\partial y} = 2$
$*$
f
14
$\frac{\partial f}{\partial f} = 1$

The function is

$$f(a, b, c) = (a + b)(b + c)$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial b} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial b} = 7 + 2 = 9$$

Gradients add at branches!

Figure 10 Backward Propagation calculations by hand[5]

Gradient Descent: Gradient descent is a process which estimates the global minima of the given function. It gives the approximate solution without considering the complex symbols used in normal equations. It is just like traversing to the bottom of the hill in the direction of steepest descent [7]. The direction is calculated by negative of the gradient. This is extremely beneficial when finding global minima for a function with a large number of parameters.

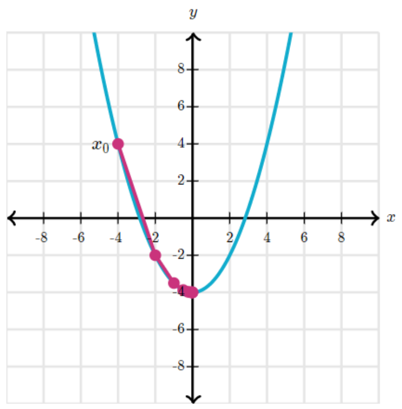


Figure 11 Gradient descent is action to find global minima [7]

When we talk about training the neural network, we are basically updating the weights of between the neurons such that given a set of input values we get the corresponding output value. Technically the Deep Neural Network the training happens through gradient decent algorithm, where the weights are updated through the process of first forward propagation and then back propagation. This one step is called an epoch. Neural Networks when training have some hyper-parameters whose values can be tuned to achieve better final accuracy. Some these are Hidden neurons per layer(neuron cells per layer), learning rate(how fast the gradient descent algorithm converges at global minima), epochs(number of training pass to be done on entire/partial dataset).

3.12 Transformer Neural Networks

Transformer Neural Networks were invented to replace the Recurrent Neural Networks(RNN's) since they could handle data only sequentially and not parallelly. Processing entire strings in parallel allowed them to take advantage of the new Graphics Processing Unit chips. This also allowed better representation of the vector space embeddings to best capture semantic representation. They too like RNN's have an encoder decoder architecture.

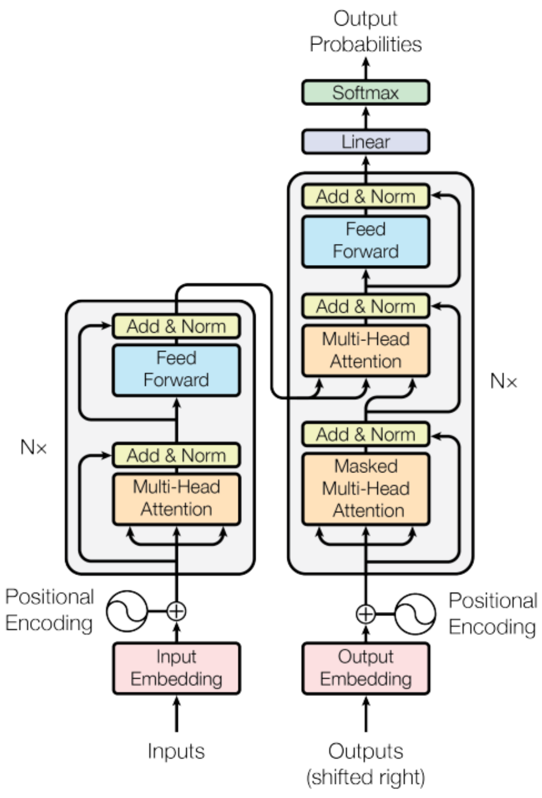


Figure 12 Transformer Architecture [8]

Now let’s take a glance at the internal working of transformers with reference to Figure 12 by taking English to French language translation example. The input sentence is “*Rutwik studies computer science.*” and translated output sentence is “*Rutwik étudie l'informatique.*”:

Encoder Block:

1. Input embeddings: This is same as vector space embeddings discussed above.
2. Positional Encoding: It creates a vector space embedding and is based on distance of words in the sentences. The original paper[8] sine and cosine function to generate this.
3. After passing the input sentence through the input embedding and applying the positional encoding we get word vectors that have positional information i.e., how each word position is related to the other words in sentence. This is then passed to the multi-headed attention layer and a feed forward layer.
4. Multi-Head Attention Layer: Attention layer determines by creating an attention vector which answers the question of which part of input it should focus on i.e., how relevant is the i^{th} word in the sentence. This is done to capture the contextual relationship between the words.
5. Feed-Forward Network: This is just a simple feedforward neural network that is applied to each of the attention vectors. The duty of the feed forward neural network is to transform the attention vectors into the form that is viable for the next encoder block or the decoder block.

Decoder Block:

1. The decoder block has three components, of which two are similar, the self-attention block which generates the attention vectors for each word to represent how much each word is related to every word in same sentence.
2. Multi-Head Attention Layer: This block can be called as encoder-decoder block since it receives inputs from both encoder and decoder. This attention block determines how each word vector from input and output are related. This is where the word mapping between input and output happens. The output of this block is a vector that represents this relationship.
3. Linear Layer: After the feed forward layer of decoder comes the linear layer. This is used to expand the dimensions into the specified length of the desired output sequence. A Softmax layer transforms this output into a probability distribution which is human interpretable. The final word is the word corresponding to the highest probability overall.

   This decoder block is run multiple times until end of sentence(EOS) is reached. Then the final vector is decoded based on vector space embeddings and presented as string as output.

3.13 Transfer Learning

Transfer learning term refers to the process of using a pretrained model created for a task is reused for another task that is similar by loading the last trained model checkpoint[9]. It is depicted in Figure 13. This is very advantages in many ways. Firstly, we don’t need to retrain the entire model on our dataset reducing significant amount of training time. Next since we don’t need to train from scratch, only minor weight update takes place and mostly the final layer is trained to adapt to the new task. This is also useful when the dataset for the new task is similar by much smaller in size.

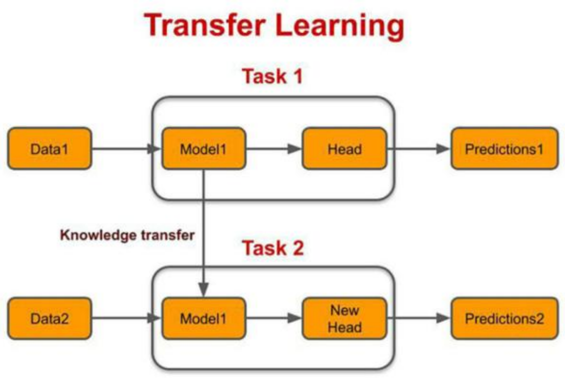


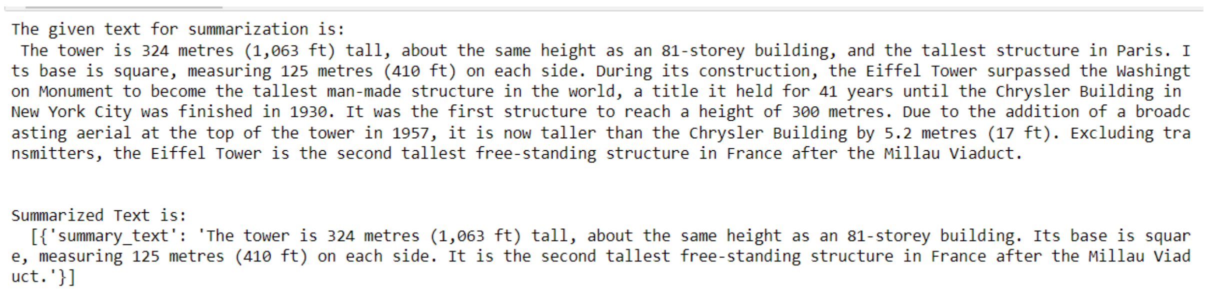
Figure 13 Transfer Learning visualized [10]

Transfer learning in Natural Language Processing has taken up pace after Hugging Face repository was launched, where thousands of base and fine-tuned NLP models are freely available. This made usage and fine turning of heavy weight models easy, fast, and provided high accuracy.

3.14 Text Summarisation

Text summarization is the process of reducing the length of a paragraph from n sentences to m sentences, where n>>m. This is actually a sequence-to-sequence generation problem for which many solutions were provided. The approach to summarization is usually divided based on the techniques used which fall within two categories Extractive or Abstractive.

Extractive summarization directly takes the most important sentences and puts it into summary[11]. Here there is no understanding of the text but just blindly copying content. In Abstractive summarization use word embeddings like that of BERT that understand semantics of language which maintains the meaning but generates some new sentences. Some of these are TextRank which uses extractive, Seq2Seq and BART are both abstractive based.



```
The given text for summarization is:
 The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris. I
ts base is square, measuring 125 metres (410 ft) on each side. During its construction, the Eiffel Tower surpassed the Washingt
on Monument to become the tallest man-made structure in the world, a title it held for 41 years until the Chrysler Building in
New York City was finished in 1930. It was the first structure to reach a height of 300 metres. Due to the addition of a broadc
asting aerial at the top of the tower in 1957, it is now taller than the Chrysler Building by 5.2 metres (17 ft). Excluding tra
nsmitters, the Eiffel Tower is the second tallest free-standing structure in France after the Millau Viaduct.


Summarized Text is:
  [{'summary_text': 'The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building. Its base is squar
e, measuring 125 metres (410 ft) on each side. It is the second tallest free-standing structure in France after the Millau Viad
uct.'}]
```

Figure 14 Text summarization using Facebook BART Large[13]

3.15 Sentiment Analysis

Humans communicate their feelings of their thoughts through sentiment. These sentiments are broadly classified into three categories, Positive, Neutral and Negative. With the rise of social media and also text generating AI, it is the need of the hour to generate an AI which can detect various sentiments of online text to regulate mediate their platforms[12].

Under the hood sentiment analysis is just a binary/multi-class classification. However, the trick lies in the input text. A lot of the above-mentioned techniques [12] can be used to create a sentiment analysis model and finally perform classification at the end to determine the sentiment. But again, the use of transformers wins the bet by huge margin due to its semantic understanding capabilities.

3.16 Blended Skill Talk

Blended Skill Talk was introduced in year 2020 in the research paper [14], which made progress about how to enable a technique to blend knowledge, empathy, and engagingness into conversational agent’s responses. A new form of dataset was required that classified these parameters and learn to generate appropriate new response.

This approach helps give the AI a persona of its own, making interactions with it unique. This project also implements Blended Skill Talk but in a different way. In the implementation section we will see how is it different and why is makes sense to have a simpler version of it for our chatbot.

3.17 Web Scrapping

Web scrapping means to extract information from the World Wide Web(WWW) through directly accessing it via software or libraries created for this purpose. There are rules and regulations to do so. The Europe is bound by GDPR(General Data Protection Regulation) and UK has UK GDPR to protect information on the internet pertaining to individuals is not misused. For the purpose of this project, we have scrapped data from .org websites which are available to public use.

Chapter IV:IMPLEMENTATION OF OPINION FORMING INTERACTIVE AI BASED ON NLP/NLU, FOCUSED ON ECONOMICS

4.1 Background and significance of Python Libraries used

This section lists the programming language and their libraries used in this project.

4.1.1 Python Programming Language

Python was introduced in 1991 by its creator Guido van Rossum. It is a high-level, general-purpose programming language[15] and has recently caught up in the Artificial Intelligence and computer engineers due to multiple machine learning and deep learning library support and easy prototyping capabilities. This entire project is built in Python.

4.1.2 NLTK

Natural Language Toolkit popularly known as NLTK is free library of basic and intermediate NLP techniques under one hood. It has several corpora and lexical resources for tasks such as bag of words, stemming, lemmatization, tokenisation, parsing and classification[16]. The community of its user and developers is strong and still used for basic tasks.

4.1.3 Sci-Kit learn

Sci-Kit learn is free library for python which has easy to use and modify machine learning algorithms built in according to the task[17].[18] It is open source providing efficient tools for predictive analysis built on NumPy, SciPy and Matplotlib.

4.1.4 Pytorch

Pytorch is an open-source deep learning framework based upon Python [19]. It is widely used by academicians, researchers, and developers since it gives precise control of deep neural net model parameters, easy modelling, and is fast, efficient in execution.

4.1.5 TensorFlow

TensorFlow is another deep learning library which is usually for deploying deep neural network models in production. It’s built around the keras framework and is spearheaded by Google. It offers support for Python and C++[20]. We use the TensorFlow framework to fine tune our sentiment analysis and summarization models.

4.1.6 HuggingFace Transformers

HuggingFace transformers library is an open-source platform hosting a collection of NLP models, datasets particularly based on Transformers[30]. Computer scientists, enthusiasts and software developers can build train and deploy their own custom models on this platform[30]. The result generation process for every model is automated, giving comparisons directly. We use these transformers models and fine tune it according to our dataset.

4.1.7 Sentence Transformers

Sentence Transformers were introduced in 2019, by Nils Reimers, Iryna Gurevych [14]and the primary reason was that when we used the traditional BERT embeddings it was not scalable and took a lot of operation to generate and compare[31]. Using sentence transformers, the latency was reduced and the word embeddings could be reused to apply custom algorithms improving their usability.

4.1.8 GUI interface with Tkinter

Tkinter is a python library which come in-built and has cross-platform functionality. It is included as standard library with python. The UI is however a bit outdated but the community around it is strong. Hence it is the most used in this project rapid prototyping and it helps keep our focus on the main algorithm. It’s easy to develop a frontend standalone application which great backend.

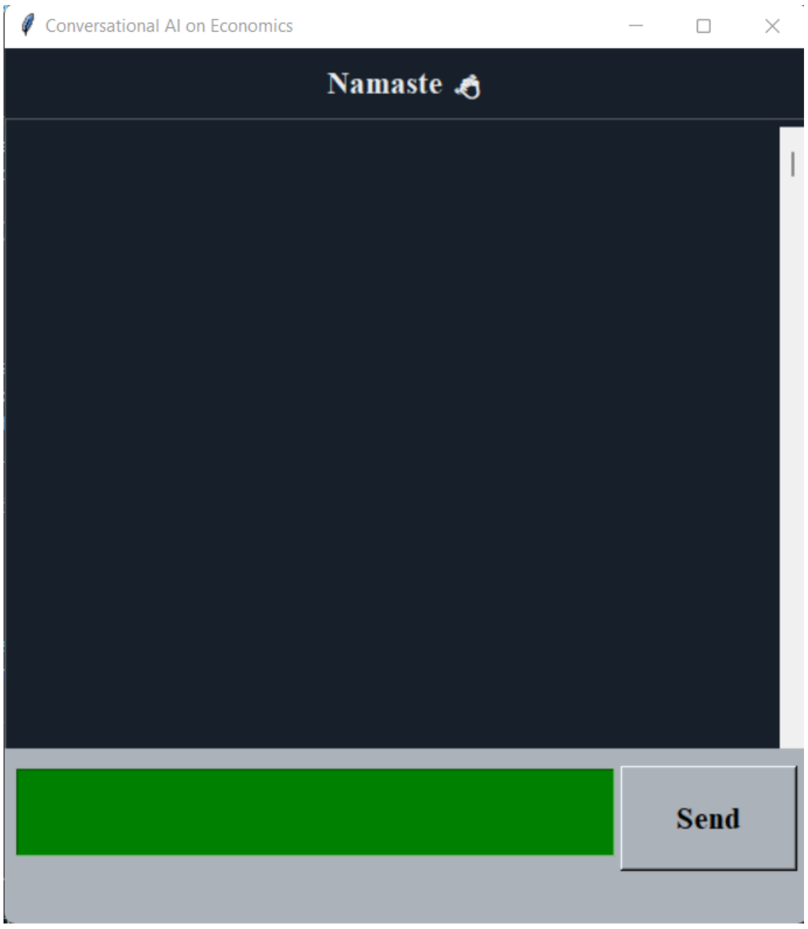


Figure 15 Present Chatbot GUI with Python Tkinter

4.2 Neural Networks and Usage

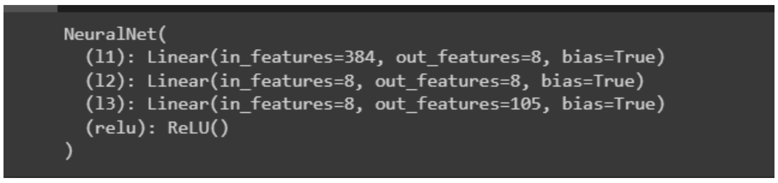
In this project we use three Neural Network for the final version of the Chat Bot. The first Neural Network is a text classification built on Pytorch. The second is a summarization model also used Pytorch. The third model is a trained for sentiment analysis and also uses Pytorch. All these neural networks are used sequentially and are called in accordance with the main algorithm.

4.2.1 Neural Network I Classification Model

This is a custom Neural Network created from scratch in Pytorch. It takes input parameters in the form of SBERT embedding. The most similar pattern identified is passed through this neural network to obtain the classified intent. Then from the response according to intent is fetched from the database and presented for further processing to Neural Networks II and III.

This Neural Network works as the information retrieval system as similar to our brain, which has structured data and we match it to some known intent(title/keyword of knowledge) which has further information on it.

The architecture of this Neural Network is as shown in Figure 16. There are 3 layers in total, each layer has 8 neurons. After layer 1 and 2 there is an activation function ‘ReLU’ as output. The last layer has the num_classes parameters set to the number of classes/intents, in our case its 105.



```
NeuralNet(
  (l1): Linear(in_features=384, out_features=8, bias=True)
  (l2): Linear(in_features=8, out_features=8, bias=True)
  (l3): Linear(in_features=8, out_features=105, bias=True)
  (relu): ReLU()
)
```

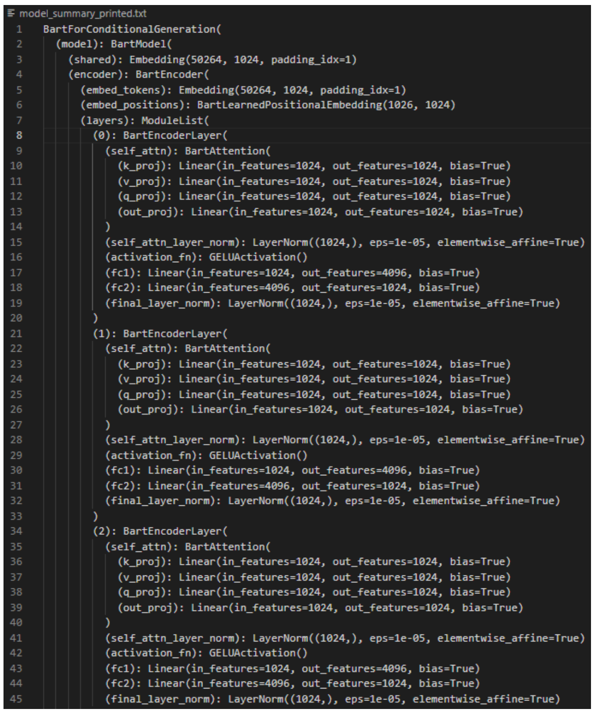
Figure 16 Architecture of Neural Network 1

4.2.2 Neural Network II Summarization Model

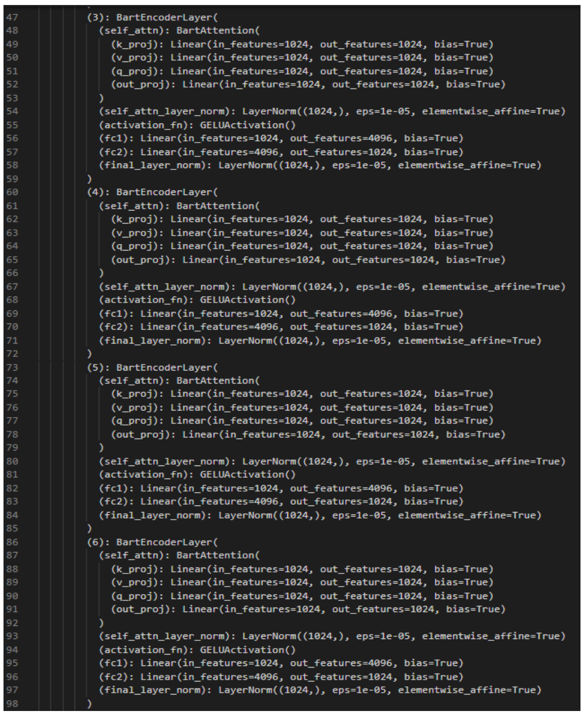
This Neural Network uses the pre-trained transformers summarization model[32] “sshleifer/distilbart-cnn-12-6”. The input to this is the retrieved response text from the knowledge database. This model generates the response summary if the response retrieved is greater than 50 words then the summary of it is returned.

This model is used to mimic how our brain retrieves information and presents it during a conversation, if the content is big and we need to simplify it in our conversation to make the listener understand better.

As we can see form the model summary shown in Figures 17-19, it has a BART architecture which is based on abstractive summarization directly. Fine tuning of this model makes no sense since our dataset size is small and is specific to basics of economics.

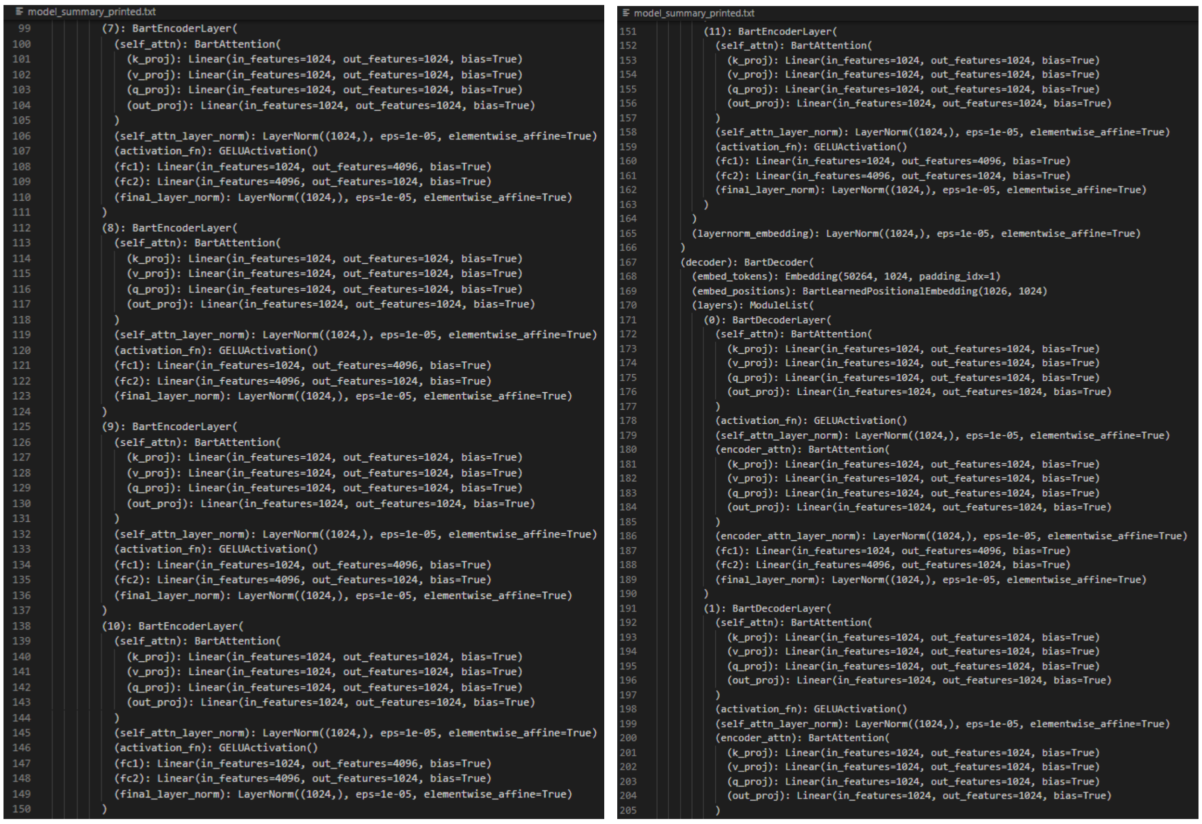


```
model_summary_printed.txt
BartForConditionalGeneration(
  (model): BartModel(
    (shared): Embedding(50264, 1024, padding_idx=1)
    (encoder): BartEncoder(
      (embed_tokens): Embedding(50264, 1024, padding_idx=1)
      (embed_positions): BartLearnedPositionalEmbedding(1026, 1024)
      (layers): ModuleList(
        (0): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (1): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (2): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
```



```
        (3): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (4): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (5): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (6): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
```

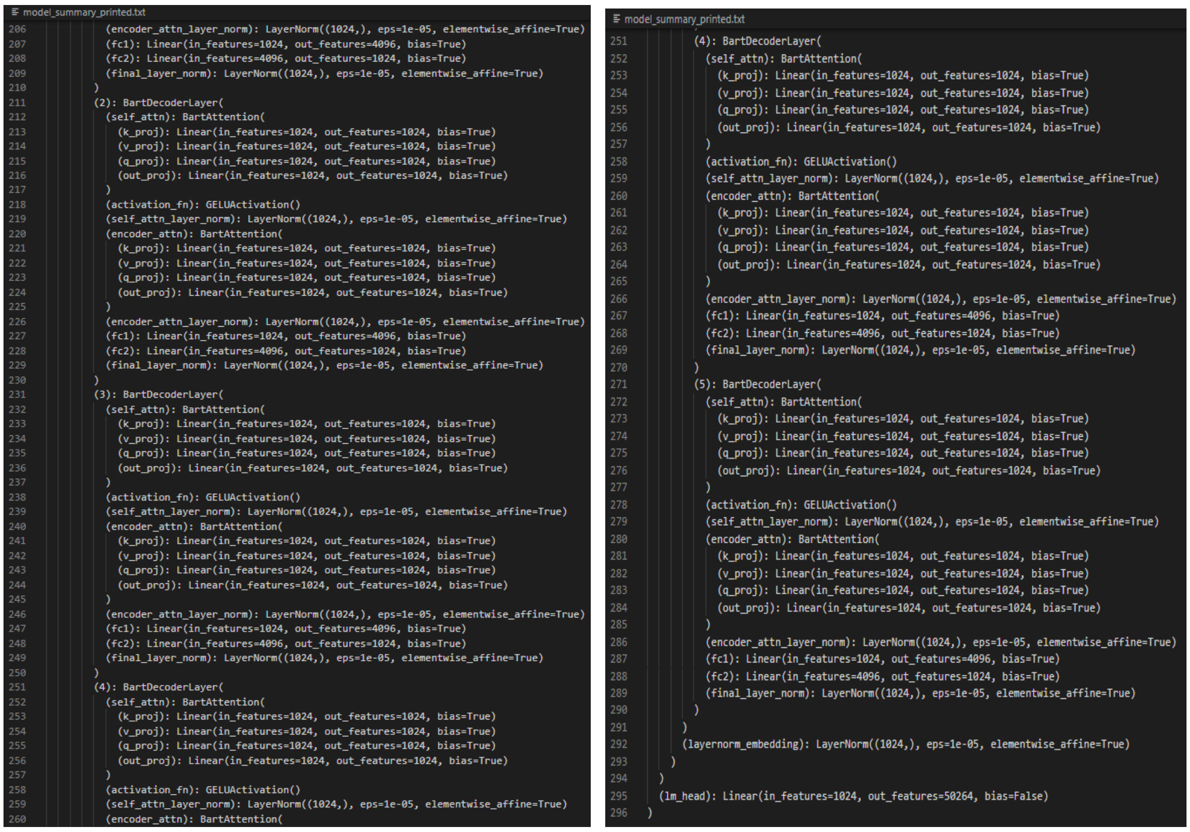
Figure 17 Summarization Model Part 1



```
        (7): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (8): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (9): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (10): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
```

```
        (11): BartEncoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (activation_fn): GELUActivation()
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
      )
      (layernorm_embedding): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
    )
    (decoder): BartDecoder(
      (embed_tokens): Embedding(50264, 1024, padding_idx=1)
      (embed_positions): BartLearnedPositionalEmbedding(1026, 1024)
      (layers): ModuleList(
        (0): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (1): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
```

Figure 18 Summarization Model Part 2



```
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (2): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (3): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (4): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
```

```
        (4): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
        (5): BartDecoderLayer(
          (self_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (activation_fn): GELUActivation()
          (self_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (encoder_attn): BartAttention(
            (k_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (v_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (q_proj): Linear(in_features=1024, out_features=1024, bias=True)
            (out_proj): Linear(in_features=1024, out_features=1024, bias=True)
          )
          (encoder_attn_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (fc1): Linear(in_features=1024, out_features=4096, bias=True)
          (fc2): Linear(in_features=4096, out_features=1024, bias=True)
          (final_layer_norm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
        )
      )
      (layernorm_embedding): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
    )
  )
  (lm_head): Linear(in_features=1024, out_features=50264, bias=False)
)
```

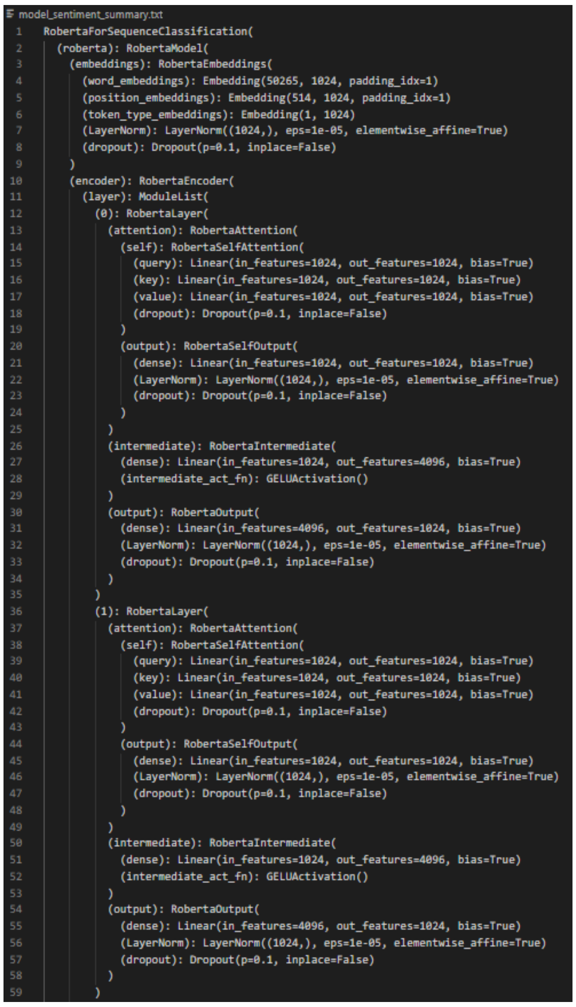
Figure 19 Summarization Model Part 3

4.2.3 Neural Network III, Sentiment Analysis

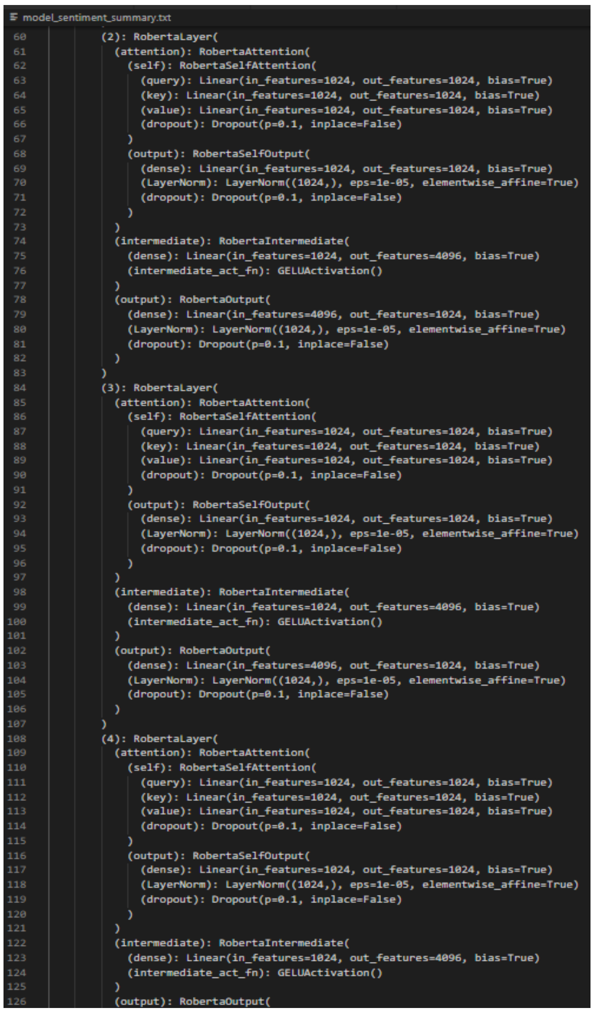
This Neural Network uses the pre-trained transformers summarization model[33] “siebert/sentiment-roberta-large-english”. The input to this is the retrieved summarized/non-summarized(depends on word size) response text from the knowledge database. This model generates the sentiment for response retrieved.

This model is used to mimic how our brain interprets the retrieved information on topic and presents it during a conversation, if the content is big and we need to simplify it in our conversation to make the listener understand better.

As we can see form the model summary shown in Figures 20-24, it has a Roberta architecture is tuned for sentiment analysis. Fine tuning of this model is not done since our dataset is small but having general English tuned improves efficiency for terms on general sense. To make this comparison another well-known model [34]“ ProsusAI/finbert” and the earlier was found to be better at interpreting topics like ‘inflation’.

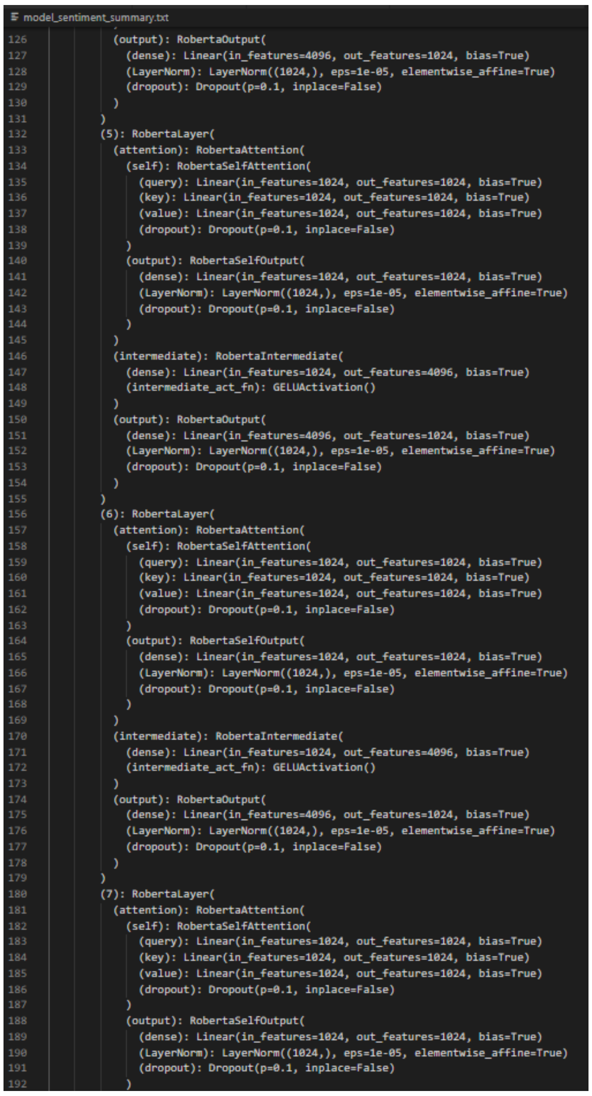

model_sentiment_summary.txt

```
RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 1024, padding_idx=1)
      (position_embeddings): Embedding(514, 1024, padding_idx=1)
      (token_type_embeddings): Embedding(1, 1024)
      (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (1): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
```


model_sentiment_summary.txt

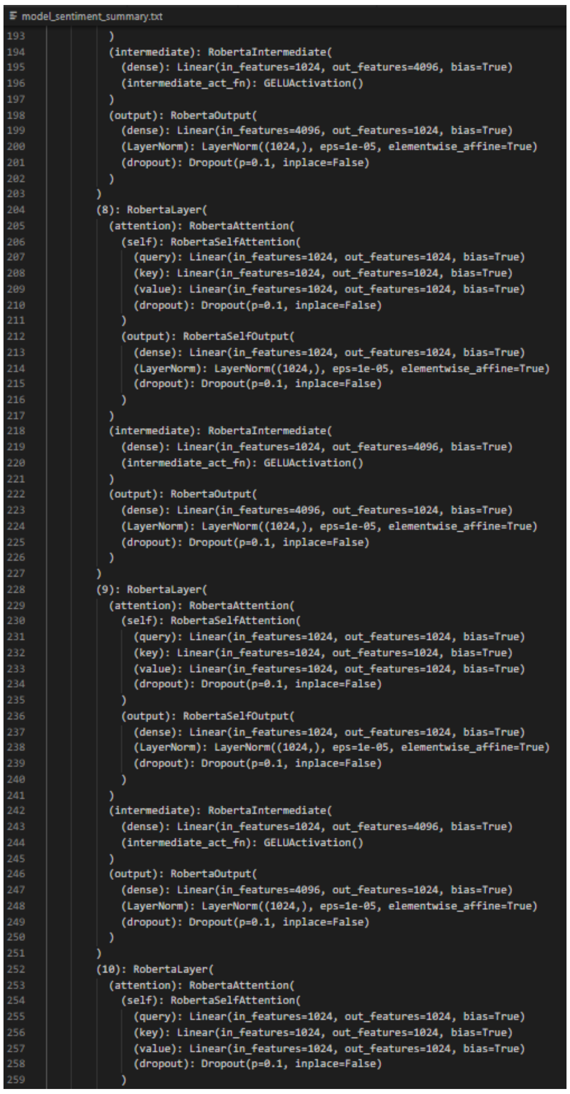
```
        (2): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (3): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (4): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
```

Figure 20 Sentiment Analysis Model Part 1



model_sentiment_summary.txt

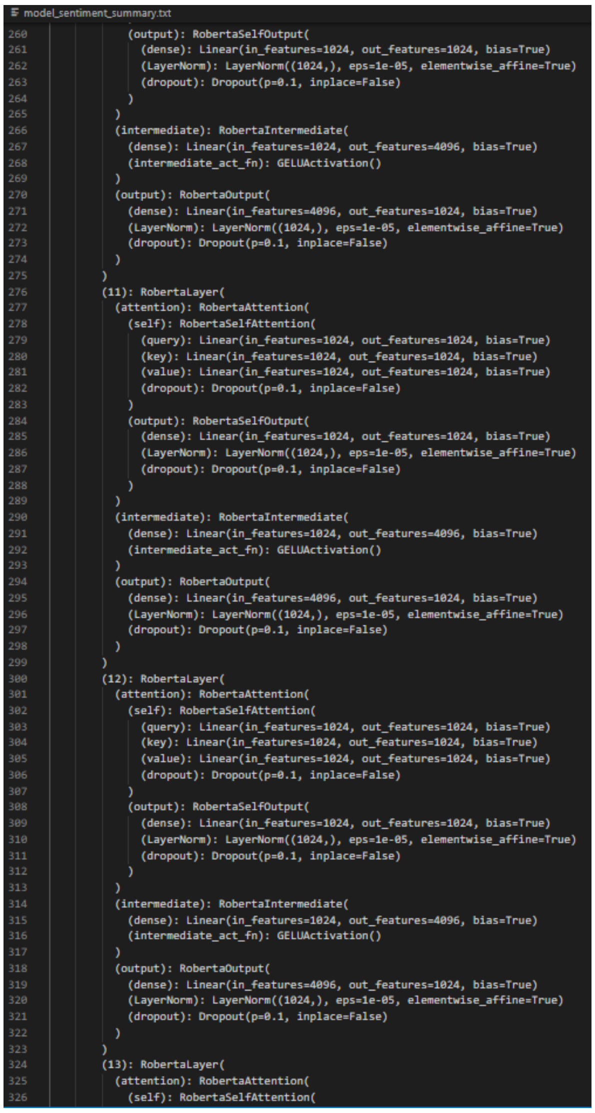
```
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (5): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (6): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (7): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
```



model_sentiment_summary.txt

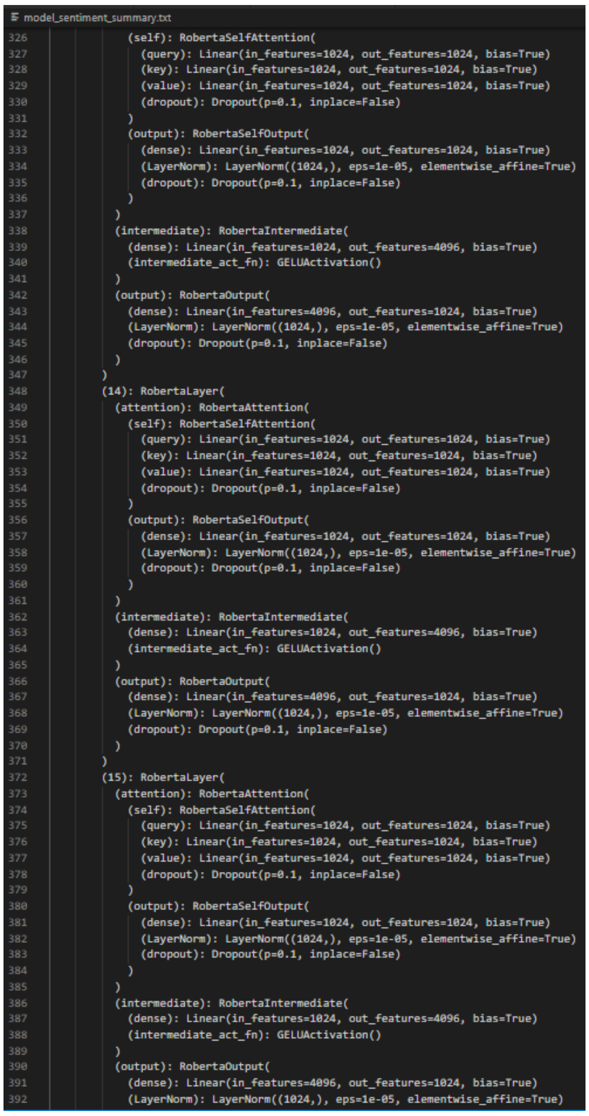
```
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (8): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (9): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (10): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
```

Figure 21 Sentiment Analysis Model Part 2



model_sentiment_summary.txt

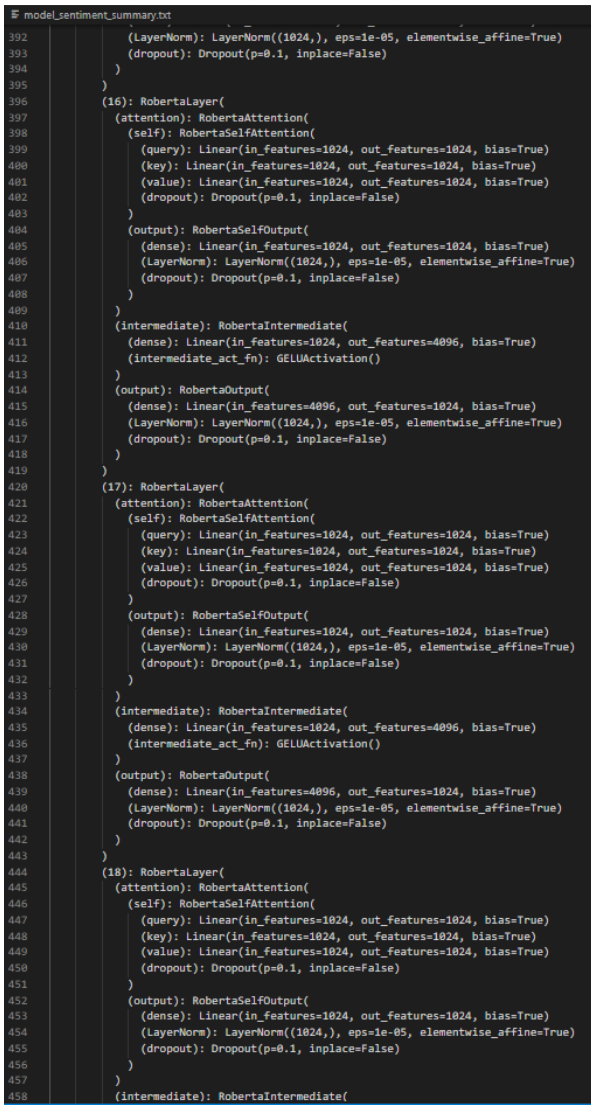
```
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (11): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (12): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (13): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
```



model_sentiment_summary.txt

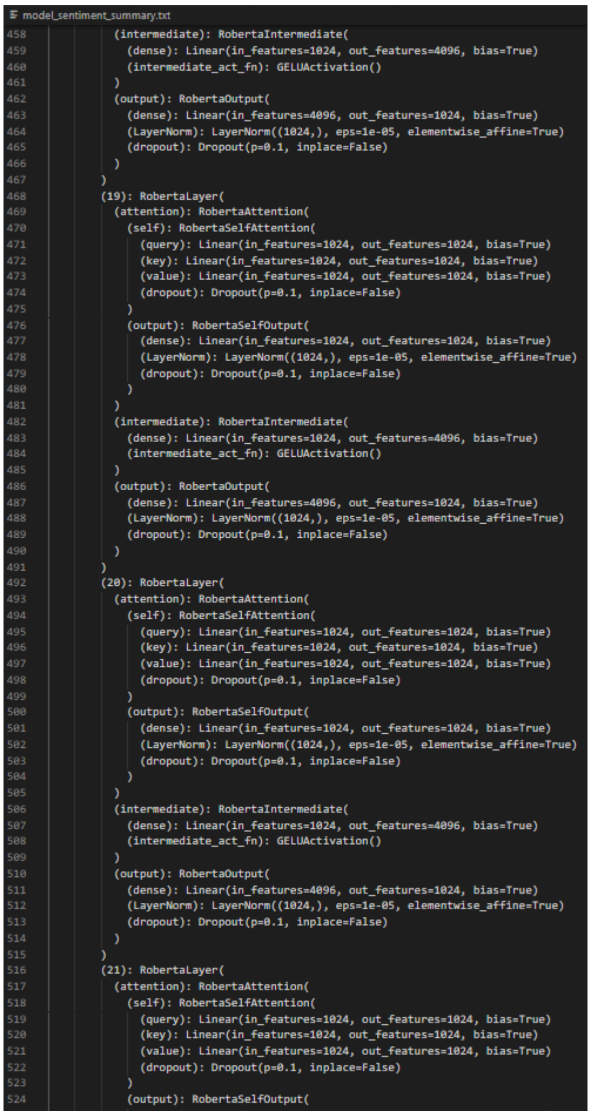
```
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (14): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (15): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
```

Figure 22 Sentiment Analysis Model Part 3



model_sentiment_summary.txt

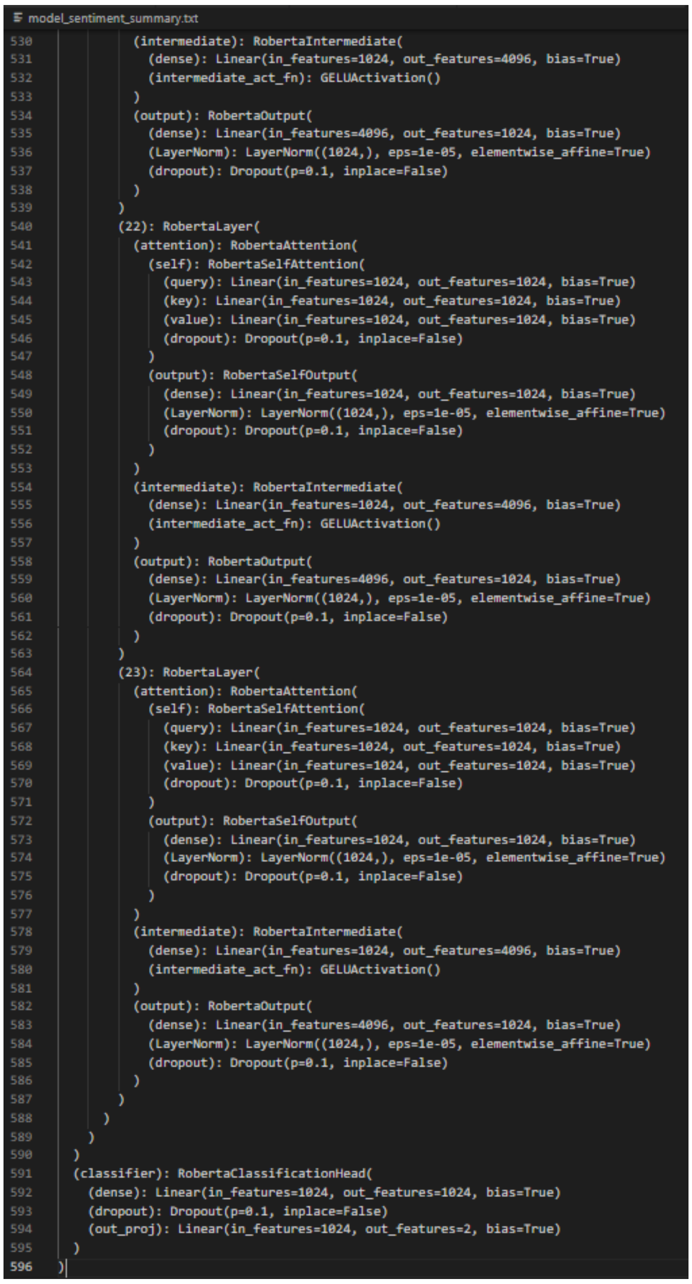
```
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (16): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (17): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (18): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
```



model_sentiment_summary.txt

```
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (19): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (20): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=1024, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=1024, out_features=4096, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=4096, out_features=1024, bias=True)
          (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (21): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=1024, out_features=1024, bias=True)
            (key): Linear(in_features=1024, out_features=1024, bias=True)
            (value): Linear(in_features=1024, out_features=1024, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
```

Figure 23 Sentiment Analysis Model Part 4



```
model_sentiment_summary.txt
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (22): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (23): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=1024, out_features=1024, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=1024, out_features=2, bias=True)
  )
)
```

Figure 24 Sentiment Analysis Model Part 5

4.3 Dataset creation

4.3.1 Web Scrapping Tool Used

For this project a web scrapping tool known as Webscrapper.io has been used. This tool allows us to scrape from permitted websites legally as stated on the website.

4.3.2 Dataset 1 creation

The dataset Data_V2.csv contains data from the book 'Economics for Dummies'. Our primary source of information was the publicly available articles of the book from Google, [36]. The scrapped articles were converted to a pdf for easier read and then converted to txt and csv format with the appropriate columns to load with the help of pandas library. This is not the main dataset and was only used for Experiment III explained in next section.

4.3.3 Dataset 2 creation

For the majority of the project that data we have used was from [39]. The information from several links to the specific topics were all downloaded at once using the Webscrapper.io tool. After this two columns namely ‘intents’ which hold the sub topics and ‘patterns’ which hold the question patters for the corresponding rows were created manually. The dataset set was then converted to the JSON format to be used as knowledge base.

4.4 Algorithm design and Experimentations in building Chatbot Application

In this sub section we will explore the initial research we conducted in making the model architecture along with the algorithmic data flow. This will give us a glance of how we arrived at the final model architecture after numerous experiments.

4.4.1 Experiment I (Basic NLP)

This experiment was performed to get a hold of implementation of basic Natural Language Processing techniques of Bag of Words, Lemmatisation, Stemming and Word embedding. No specific dataset was used in this experimentation since it was initial experiment. This code was the building block for the basic Chatbot built in Experiment II and V.

4.4.2 Algorithm I

After performing experiments, I and II, we made our initial algorithm and dataflow chart for our chatbot which is shown in Figure 25. Let’s take a deeper look at it.

Firstly, we get the query in the form of question from the user. Then we use tokenize the sentence into words, apply stemming to reduce each word to its roots, finally creating a vector representation of this tokenized sentence with the bag of words corpus dictionary. This previous step is also applied on the pattern questions from the database. Finally, cosine similarity(distance metric) is used to calculate the nearest vector to user query in stored pattern questions.

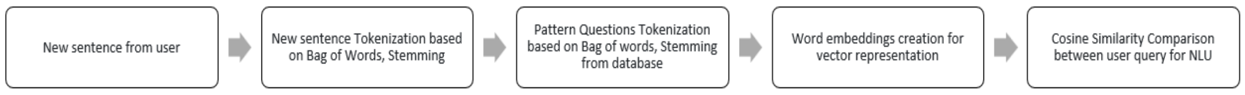

New sentence from user
New sentence Tokenization based on Bag of Words, Stemming
Pattern Questions Tokenization based on Bag of words, Stemming from database
Word embeddings creation for vector representation
Cosine Similarity Comparison between user query for NLU

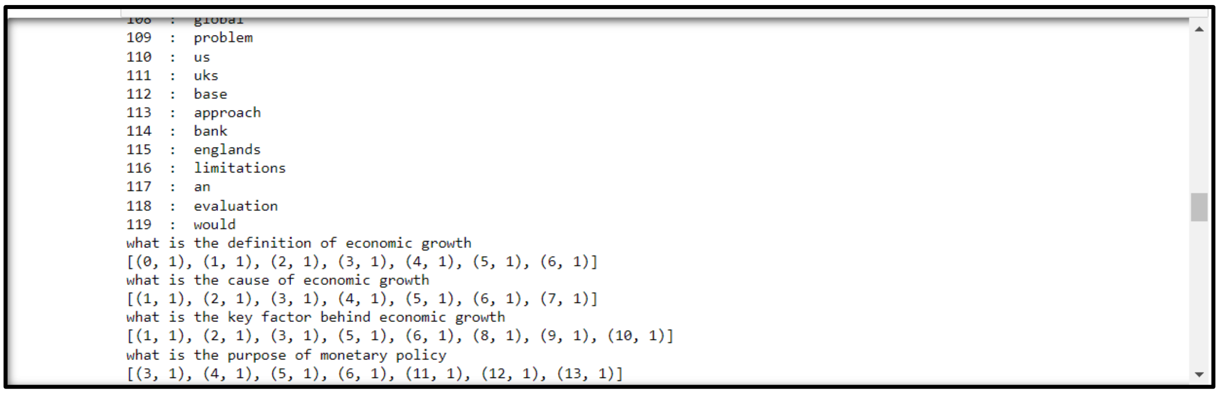
Figure 25 Algorithm I Flow Chart

4.4.3 Experiment II (Cosine Similarity and Sample Question Answering)

As we started building dataset, we need to get user query and match it against the question pattern to identify the topic to retrieve the information from our created database. In this experiment we will see how we can achieve this.

This experiment is divided into two parts, let’s take a closer look at their implementation.

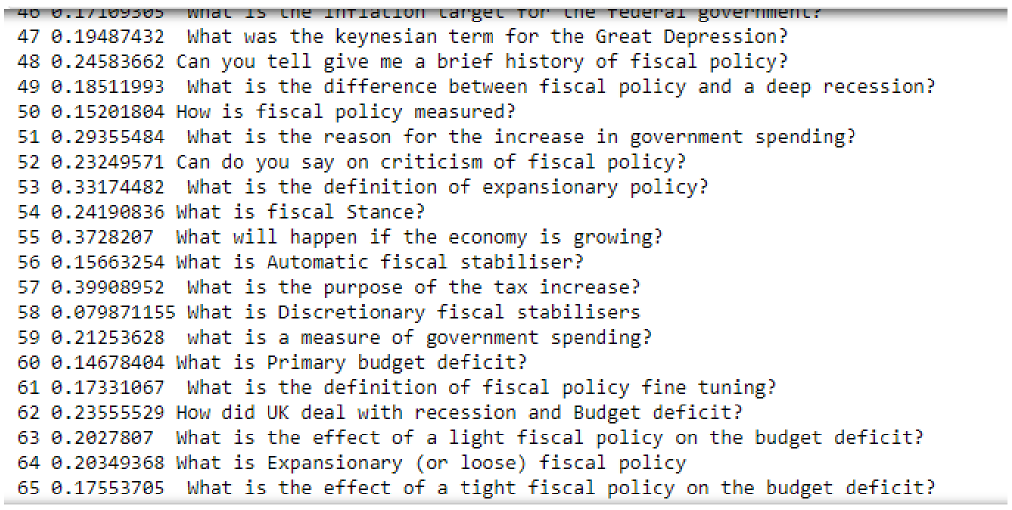
PART 1: By utilising the Bag of Words, our first approach uses semantic similarity. Each phrase is encoded into a vector by Bag of Words, and the vector's length equals the vocabulary's total number of words. The vector's elements each show the frequency of a certain word in the given text. The dictionary and the responses retrieved are printed in the sparse manner as a sample as shown in Figure 26. Then we use cosine similarity to find the distance between the vectors of the original query and all the question patterns. The best matching i.e., the vectors with highest similarity are returned.



```
108  :  global
109  :  problem
110  :  us
111  :  uks
112  :  base
113  :  approach
114  :  bank
115  :  englands
116  :  limitations
117  :  an
118  :  evaluation
119  :  would
what is the definition of economic growth
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1)]
what is the cause of economic growth
[(1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)]
what is the key factor behind economic growth
[(1, 1), (2, 1), (3, 1), (5, 1), (6, 1), (8, 1), (9, 1), (10, 1)]
what is the purpose of monetary policy
[(3, 1), (4, 1), (5, 1), (6, 1), (11, 1), (12, 1), (13, 1)]
```

Figure 26 Bag of words approach for question patterns in dataset

PART 2: In this part we fix the semantic information not being preserved. For this we use the SBERT model 'sentence-transformers/all-MiniLM-L6-v2' to create the word embeddings. We convert our original query and all the question patterns into SBERT embeddings. Then we use this vector representation to find the cosine similarity to find the distance between the vectors of the original query and all the question patterns. The best match i.e., the vectors with highest similarity is returned.



```
46 0.17109305  What is the inflation target for the federal government?
47 0.19487432  What was the keynesian term for the Great Depression?
48 0.24583662 Can you tell give me a brief history of fiscal policy?
49 0.18511993  What is the difference between fiscal policy and a deep recession?
50 0.15201804 How is fiscal policy measured?
51 0.29355484  What is the reason for the increase in government spending?
52 0.23249571 Can do you say on criticism of fiscal policy?
53 0.33174482  What is the definition of expansionary policy?
54 0.24190836 What is fiscal Stance?
55 0.3728207  What will happen if the economy is growing?
56 0.15663254 What is Automatic fiscal stabiliser?
57 0.39908952  What is the purpose of the tax increase?
58 0.079871155 What is Discretionary fiscal stabilisers
59 0.21253628  what is a measure of government spending?
60 0.14678404 What is Primary budget deficit?
61 0.17331067  What is the definition of fiscal policy fine tuning?
62 0.23555529 How did UK deal with recession and Budget deficit?
63 0.2027807  What is the effect of a light fiscal policy on the budget deficit?
64 0.20349368 What is Expansionary (or loose) fiscal policy
65 0.17553705  What is the effect of a tight fiscal policy on the budget deficit?
```

Figure 27 SBERT embeddings for question patterns in dataset

4.4.4 Algorithm II

The motivation for this algorithm was to feed the text related to a topic to the trained text generation system and get the generated text paragraph based on few initial input words.

Usually, we humans read a whole textbook on a topic, and we try to retrieve based on keywords. The same thought process goes when designing this algorithm.

In this algorithm firstly we feed the entire dataset 'Data_V2.csv' consisting of rows of sentences from scrapped from the book 'Economics for Dummies' belonging to various chapters. Now a user query is taken on the and then it is stripped to get the keywords into a statement like 'Globalisation is', 'Economics history is'. Then in the next step this phrase 'Economic history is' is passed to the Sequence generation Transformer model 1 as input. The model after execution will return a sequence related to the topic of query randomly to make it look authentic. Moving forward with this output we will use it perform sentiment analysis of it. According to the sentiment of the generated text from Transformer model 2, we then use Blended Skill Talk to present these outputs in the form of conversation directly.

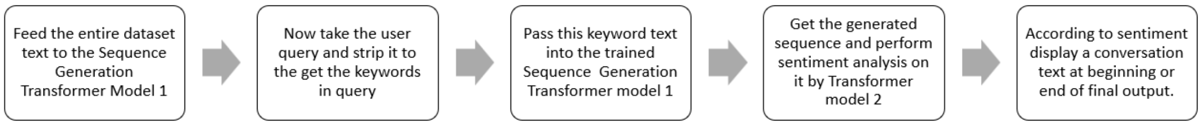
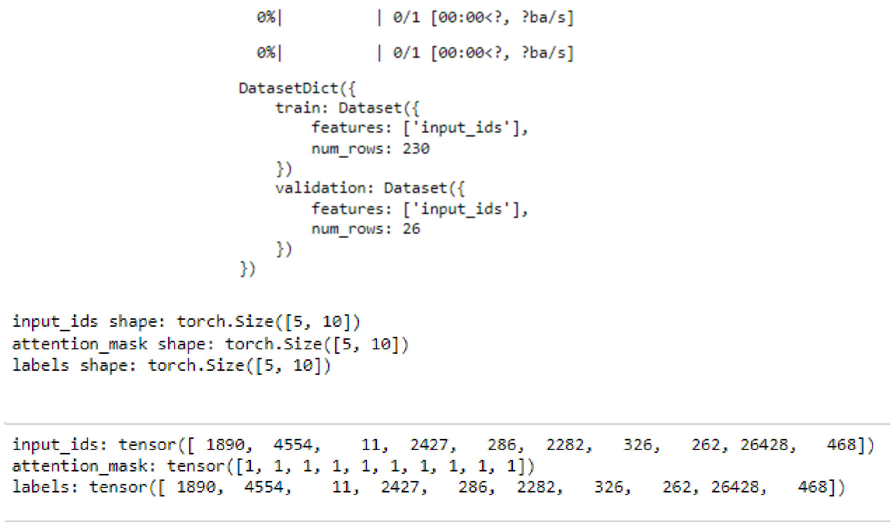

Feed the entire dataset text to the Sequence Generation Transformer Model 1
Now take the user query and strip it to the get the keywords in query
Pass this keyword text into the trained Sequence Generation Transformer model 1
Get the generated sequence and perform sentiment analysis on it by Transformer model 2
According to sentiment display a conversation text at beginning or end of final output.

Figure 28 Algorithm II Text generation with sentiment analysis

4.4.5 Experiment III (Text Generation)

In this experiment we try out the Algorithm II proposed.

In the very beginning, we load the entire dataset 'Data_V2.csv' consisting of 256 rows of sentences web scrapped from online articles 'Economics for Dummies' belonging to beginning chapters, via the load_dataset function and create a preliminary dataset. The dataset 'Data_V2.csv' was prepared for this model specifically. Then we do some text pre-processing to get the text length. This s where we tokenize it and convert it into TensorFlow dataset format.



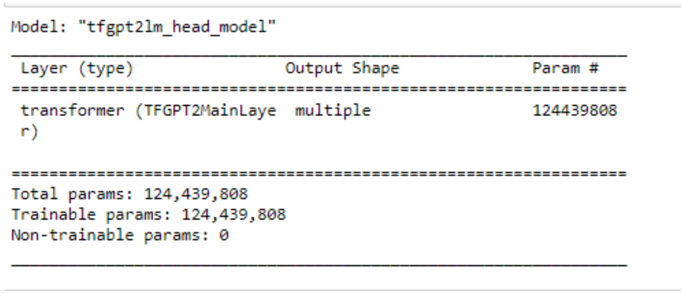
```
  0%|          | 0/1 [00:00<?, ?ba/s]

  0%|          | 0/1 [00:00<?, ?ba/s]

DatasetDict({
    train: Dataset({
        features: ['input_ids'],
        num_rows: 230
    })
    validation: Dataset({
        features: ['input_ids'],
        num_rows: 26
    })
})
```

```
input_ids shape: torch.Size([5, 10])
attention_mask shape: torch.Size([5, 10])
labels shape: torch.Size([5, 10])
```

```
input_ids: tensor([ 1890,  4554,    11,  2427,   286,  2282,   326,   262, 26428,   468])
attention_mask: tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
labels: tensor([ 1890,  4554,    11,  2427,   286,  2282,   326,   262, 26428,   468])
```

Figure 29 Word embeddings dataset with attention mask and input id shapes

The transformer model used is "GPT2"[35], a very famous architecture for several language and vision tasks. The catch for training of the Training a sequence generating model is that X_train and Y_train are the same sequences. Then we load the tokenizer for 'GPT2'.



```
Model: "tfgpt2lm_head_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 transformer (TFGPT2MainLaye  multiple                 124439808
 r)

=================================================================
Total params: 124,439,808
Trainable params: 124,439,808
Non-trainable params: 0
_________________________________________________________________
```

Figure 30 GPT-2 model parameters

We configure the hyperparameters(learning rate=5e-5, weight decay rate=0.01, warmup steps=1000) and train the model for about 20 epochs and get a loss of 9.11. Then after this we pass a query of user like 'Economic history is' to the trained model and generate the output text sequence. Then we perform the sentiment analysis of this output text. We end the result here due to unfavourable results which is discussed in Chapter V.

4.4.6 Experiment IV (Transformer based text classification)

This experiment was done to have some knowledge on how to train a classification system using transformers. The dataset used was Kaggle Fake News Dataset [40]. It uses the “*distilbert-base-uncased-finetuned-sst-2-english”[*41] model from the HuggingFace transformer library. The dataset is loaded, then the TensorFlow training dataset is created following the method mention in Experiment III. Now the hyperparameters used for training are warmup steps=50, weight decay=0.01, epochs=5, evaluation steps=10.

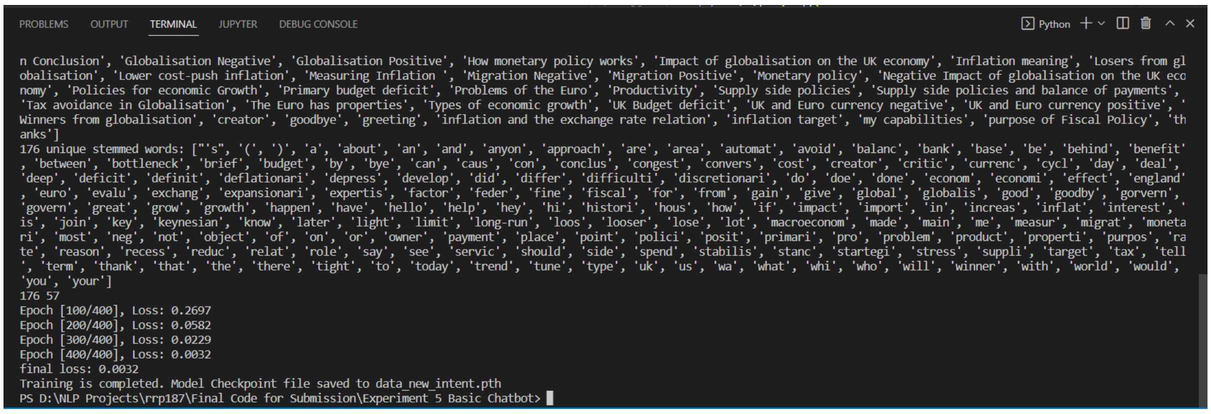
This model gives a very good accuracy of 96.39. This same approach is further used in Experiment V for sentiment analysis since both are binary classification problems.

4.4.7 Experiment V (Chatbot with Basic NLP)

This experiment directly inherits the four experiments performed above putting them all together to create our first Chat Bot Application. This is a basic user query matching and an information retrieval system similar to some used in websites since it is a light weight to train and implement.

There are five code files that are of major importance. Data is stored in the new_intents file. The NeuralNet.py file defines the neural network structure. The train.py is called to train the classification Neural Network on dataset. The inputs are the question pattern processed into vectors by Bag of Words and Stemming by the custom functions written into the nltk_all.py. Next there is bot_v2.py app which loads the trained neural network for prediction and retrieval of response based on user query. This file forms the backbone of the Chatbot application. Then finally BotApplication.py file defines the UI to be created and acts as frontend.

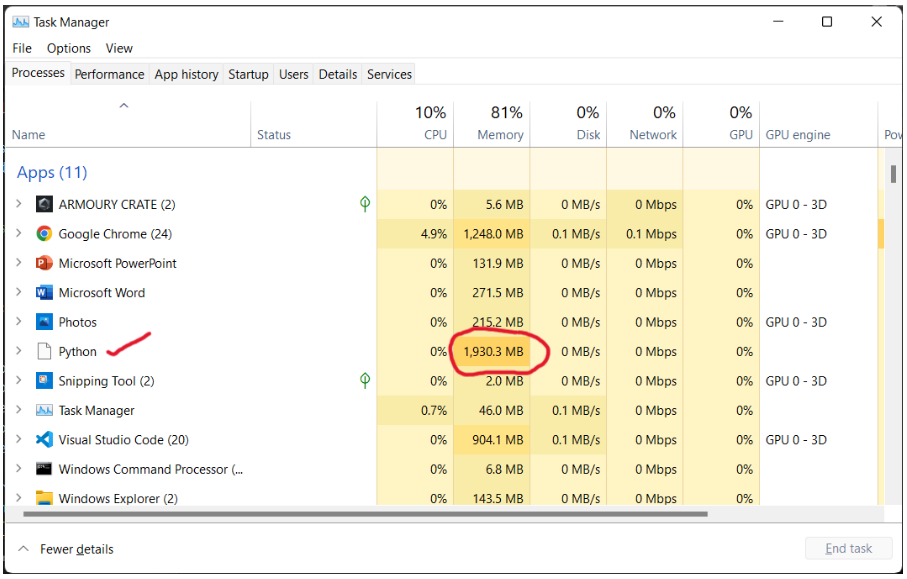
The training shown in Figure 31. The vectorized pattern are trained as inputs and intents(sub-topics) for response is the output. As we can see from the figure it takes around 400 epochs to fit. The model hyperparameters are as follows: Batch Size = 8, Learning rate = 0.001, hidden size/neurons per layer = 8.



```
n Conclusion', 'Globalisation Negative', 'Globalisation Positive', 'How monetary policy works', 'Impact of globalisation on the UK economy', 'Inflation meaning', 'Losers from gl
obalisation', 'Lower cost-push inflation', 'Measuring Inflation ', 'Migration Negative', 'Migration Positive', 'Monetary policy', 'Negative Impact of globalisation on the UK eco
nomy', 'Policies for economic Growth', 'Primary budget deficit', 'Problems of the Euro', 'Productivity', 'Supply side policies', 'Supply side policies and balance of payments',
'Tax avoidance in Globalisation', 'The Euro has properties', 'Types of economic growth', 'UK Budget deficit', 'UK and Euro currency negative', 'UK and Euro currency positive', '
Winners from globalisation', 'creator', 'goodbye', 'greeting', 'inflation and the exchange rate relation', 'inflation target', 'my capabilities', 'purpose of Fiscal Policy', 'th
anks']
176 unique stemmed words: ["'s", '(', ')', 'a', 'about', 'an', 'and', 'anyon', 'approach', 'are', 'area', 'automat', 'avoid', 'balanc', 'bank', 'base', 'be', 'behind', 'benefit'
, 'between', 'bottleneck', 'brief', 'budget', 'by', 'bye', 'can', 'caus', 'con', 'conclus', 'congest', 'convers', 'cost', 'creator', 'critic', 'currenc', 'cycl', 'day', 'deal',
'deep', 'deficit', 'definit', 'deflationari', 'depress', 'develop', 'did', 'differ', 'difficulti', 'discretionari', 'do', 'doe', 'done', 'econom', 'economi', 'effect', 'england'
, 'euro', 'evalu', 'exchang', 'expansionari', 'expertis', 'factor', 'feder', 'fine', 'fiscal', 'for', 'from', 'gain', 'give', 'global', 'globalis', 'good', 'goodby', 'gorvern',
'govern', 'great', 'grow', 'growth', 'happen', 'have', 'hello', 'help', 'hey', 'hi', 'histori', 'hous', 'how', 'if', 'impact', 'import', 'in', 'increas', 'inflat', 'interest', '
is', 'join', 'key', 'keynesian', 'know', 'later', 'light', 'limit', 'long-run', 'loos', 'looser', 'lose', 'lot', 'macroeconom', 'made', 'main', 'me', 'measur', 'migrat', 'moneta
ri', 'most', 'neg', 'not', 'object', 'of', 'on', 'or', 'owner', 'payment', 'place', 'point', 'polici', 'posit', 'primari', 'pro', 'problem', 'product', 'properti', 'purpos', 'ra
te', 'reason', 'recess', 'reduc', 'relat', 'role', 'say', 'see', 'servic', 'should', 'side', 'spend', 'stabilis', 'stanc', 'startegi', 'stress', 'suppli', 'target', 'tax', 'tell
', 'term', 'thank', 'that', 'the', 'there', 'tight', 'to', 'today', 'trend', 'tune', 'type', 'uk', 'us', 'wa', 'what', 'whi', 'who', 'will', 'winner', 'with', 'world', 'would',
'you', 'your']
176 57
Epoch [100/400], Loss: 0.2697
Epoch [200/400], Loss: 0.0582
Epoch [300/400], Loss: 0.0229
Epoch [400/400], Loss: 0.0032
final loss: 0.0032
Training is completed. Model Checkpoint file saved to data_new_intent.pth
PS D:\NLP Projects\rrp187\Final Code for Submission\Experiment 5 Basic Chatbot>
```

Figure 31 Experiment V Training of Neural Network

Performance wise the chat bot application takes 1.9GB to 2.1GB RAM to run along with ~800MB of GPU VRAM as shown in Figure 32 and 33.


Task Manager

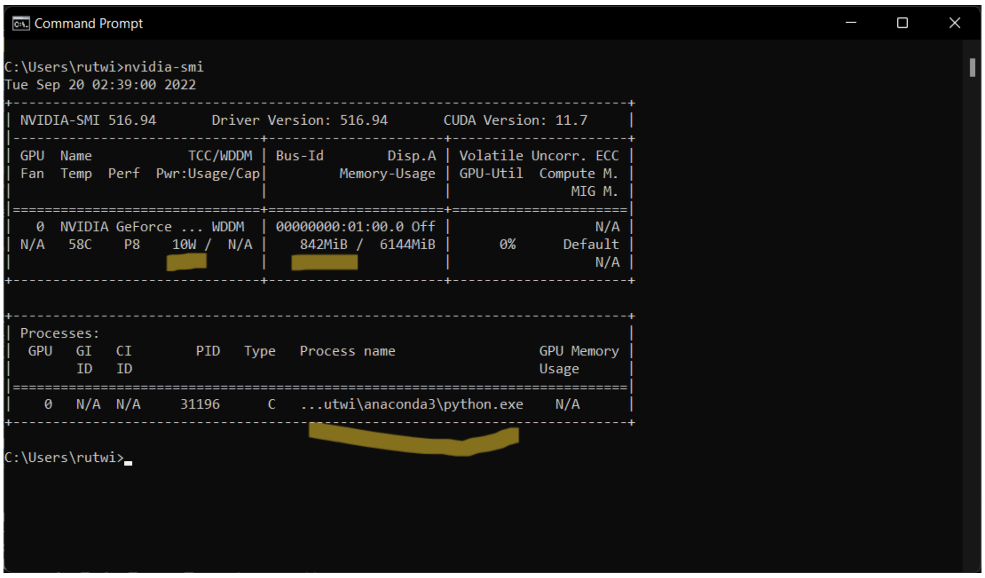
File Options View

Processes Performance App history Startup Users Details Services

Name	Status	10% CPU	81% Memory	0% Disk	0% Network	0% GPU	GPU engine
Apps (11)							
ARMOURY CRATE (2)		0%	5.6 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Google Chrome (24)		4.9%	1,248.0 MB	0.1 MB/s	0.1 Mbps	0%	GPU 0 - 3D
Microsoft PowerPoint		0%	131.9 MB	0 MB/s	0 Mbps	0%	
Microsoft Word		0%	271.5 MB	0 MB/s	0 Mbps	0%	
Photos		0%	215.2 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Python		0%	1,930.3 MB	0 MB/s	0 Mbps	0%	
Snipping Tool (2)		0%	2.0 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Task Manager		0.7%	46.0 MB	0.1 MB/s	0 Mbps	0%	
Visual Studio Code (20)		0%	904.1 MB	0.1 MB/s	0 Mbps	0%	GPU 0 - 3D
Windows Command Processor (...		0%	6.8 MB	0 MB/s	0 Mbps	0%	
Windows Explorer (2)		0%	143.5 MB	0 MB/s	0 Mbps	0%	

Fewer details End task

Figure 32 Experiment V Chat Bot total RAM Usage


Command Prompt

```
C:\Users\rutwi>nvidia-smi
Tue Sep 20 02:39:00 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 516.94       Driver Version: 516.94       CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   58C    P8    10W /  N/A |    842MiB /  6144MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     31196      C   ...utwi\anaconda3\python.exe    N/A      |
+-----------------------------------------------------------------------------+

C:\Users\rutwi>_
```

Figure 33 Experiment V Chat Bot total GPU memory usage

4.4.8 Algorithm III and Experiment VI (Opinion Forming Chatbot)

This is the final algorithm designed after performing the Experiments I-V. Using this algorithm shown in figure 34, the final Opinion forming Chat Bot on Economics was designed.

The submission for this experiment has four major files. First one is NeuralNet.py which has the code for structure of Neural Network and Train.py file which is same as the one in previous experiment. The file named sim_test_senti.py has the code flow for the algorithm III. The hyperparameters for Neural Network I remain same as per the previous experiment. We run the Chatbot application by running bot_app.py file.

Let's walk through the algorithm in detail.

Firstly, we get the user query through via the chatbot application interface. Then we convert the input into vector embeddings using BERT encodings. Then we do a sentence similarity between user input to the stored question patterns. After getting a match we retrieve the response using Neural Network 1. We then perform retrieved response summarization if the text is too long using Neural Network 2. In the second last step of our processing, we perform a sentiment analysis on the summarized/retrieved knowledge by Neural Network 3.

Finally, all the outputs are put together by using Blended Skill Talk technique to display output in Chatbot window.

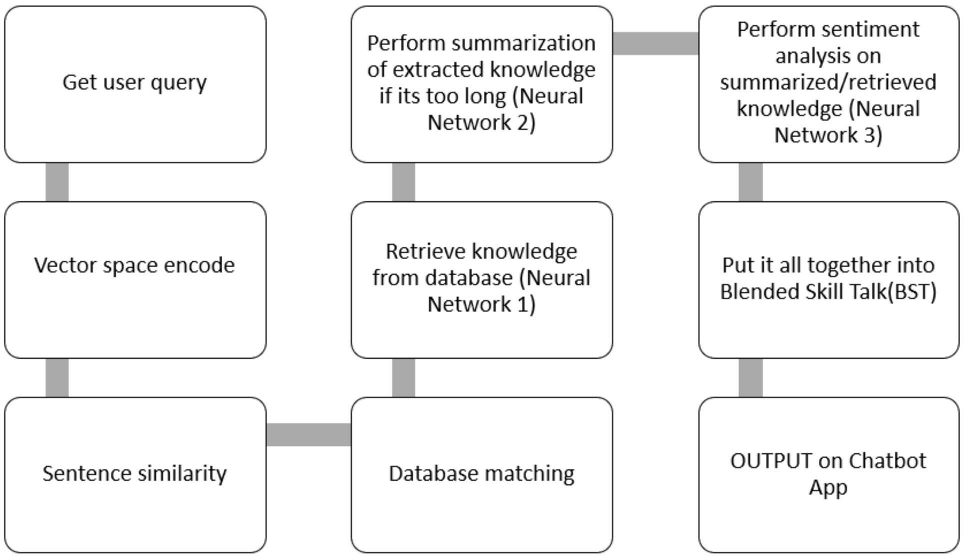
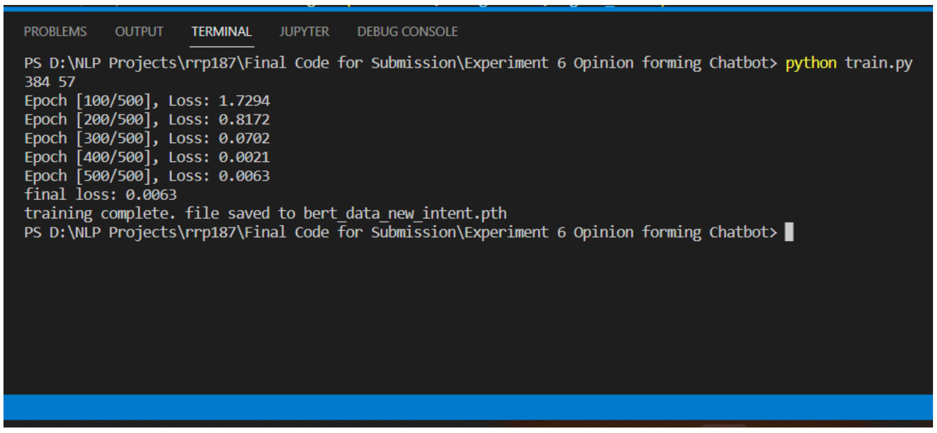

Get user query
Vector space encode
Sentence similarity
Database matching
Retrieve knowledge from database (Neural Network 1)
Perform summarization of extracted knowledge if its too long (Neural Network 2)
Perform sentiment analysis on summarized/retrieved knowledge (Neural Network 3)
Put it all together into Blended Skill Talk(BST)
OUTPUT on Chatbot App

Figure 34 Algorithm III Flow Chart

To implement blended skill talk we have created three text lists namely, start response, negative phrases and positive phrases of which the latter two are used to convey opinions response. These lists contain the text responses needed for the conversational AI to skilfully portray its speech to user. The recipe developed for output response of the AI here is:

Random Start response + "Summarized/Extracted text" + Random opinion response

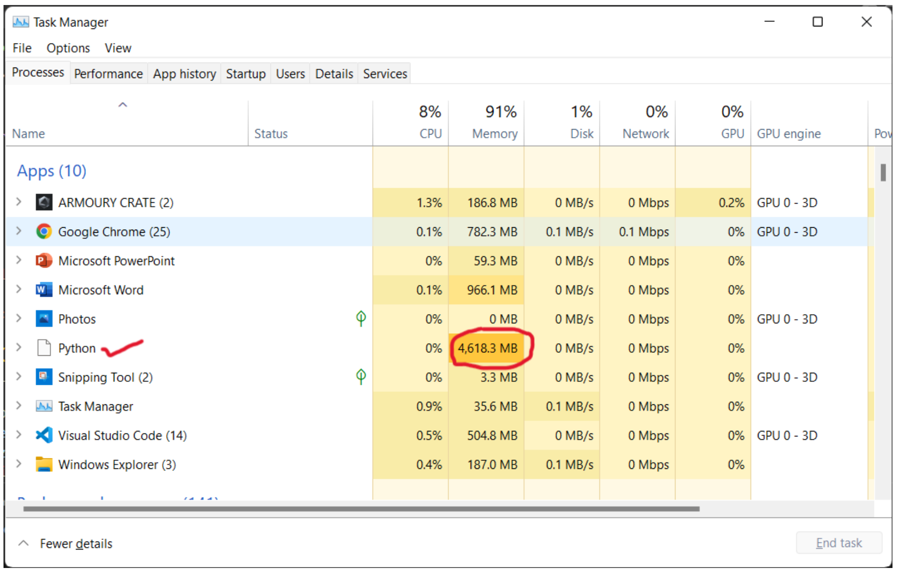
The training shown in Figure .



```
PS D:\NLP Projects\rrp187\Final Code for Submission\Experiment 6 Opinion forming Chatbot> python train.py
384 57
Epoch [100/500], Loss: 1.7294
Epoch [200/500], Loss: 0.8172
Epoch [300/500], Loss: 0.0702
Epoch [400/500], Loss: 0.0021
Epoch [500/500], Loss: 0.0063
final loss: 0.0063
training complete. file saved to bert_data_new_intent.pth
PS D:\NLP Projects\rrp187\Final Code for Submission\Experiment 6 Opinion forming Chatbot>
```

Figure 35 Experiment VI Training

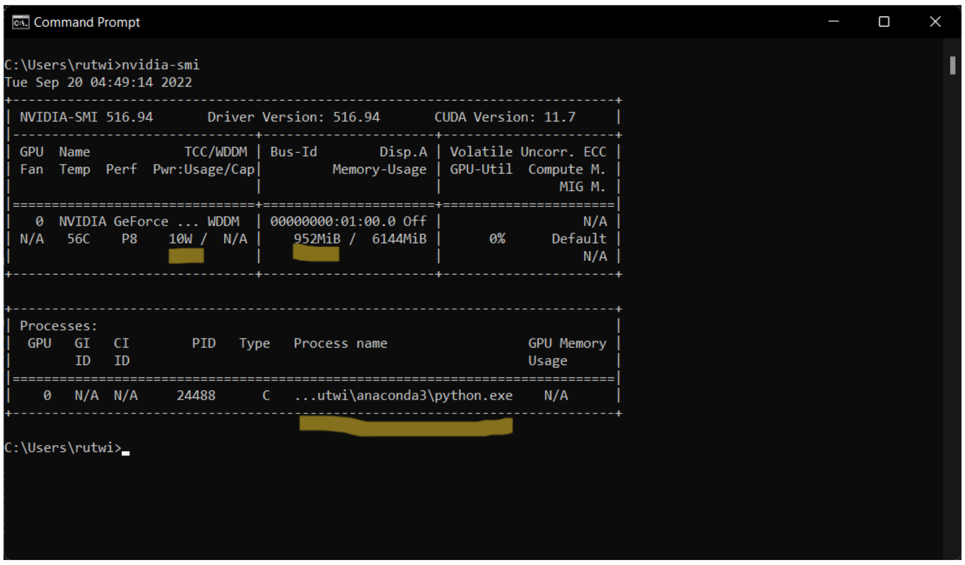
Performance wise the chat bot application takes around 4.7GB RAM to run along with ~900MB of GPU VRAM as shown in Figure 36 and 37.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	Status	8% CPU	91% Memory	1% Disk	0% Network	0% GPU	GPU engine
Apps (10)							
ARMOURY CRATE (2)		1.3%	186.8 MB	0 MB/s	0 Mbps	0.2%	GPU 0 - 3D
Google Chrome (25)		0.1%	782.3 MB	0.1 MB/s	0.1 Mbps	0%	GPU 0 - 3D
Microsoft PowerPoint		0%	59.3 MB	0 MB/s	0 Mbps	0%	
Microsoft Word		0.1%	966.1 MB	0 MB/s	0 Mbps	0%	
Photos		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Python		0%	4,618.3 MB	0 MB/s	0 Mbps	0%	
Snipping Tool (2)		0%	3.3 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Task Manager		0.9%	35.6 MB	0.1 MB/s	0 Mbps	0%	
Visual Studio Code (14)		0.5%	504.8 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Windows Explorer (3)		0.4%	187.0 MB	0.1 MB/s	0 Mbps	0%	

Fewer details   End task

Figure 36 Experiment VI Chat Bot total RAM Usage


Command Prompt

```
C:\Users\rutwi>nvidia-smi
Tue Sep 20 04:49:14 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 516.94       Driver Version: 516.94       CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   56C    P8    10W /  N/A |    952MiB /  6144MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     24488      C   ...utwi\anaconda3\python.exe    N/A      |
+-----------------------------------------------------------------------------+

C:\Users\rutwi>
```

Figure 37 Experiment VI Chat Bot total GPU memory usage

Chapter V: RESULTS AND DISCUSSION

Introduction

In this section we will see the results of our experiments and the challenges faced in all the approaches. We will also address these issues and suggest further improvements. We will also have a discussion on these approaches.

5.1 Experiment I (Basic NLP)

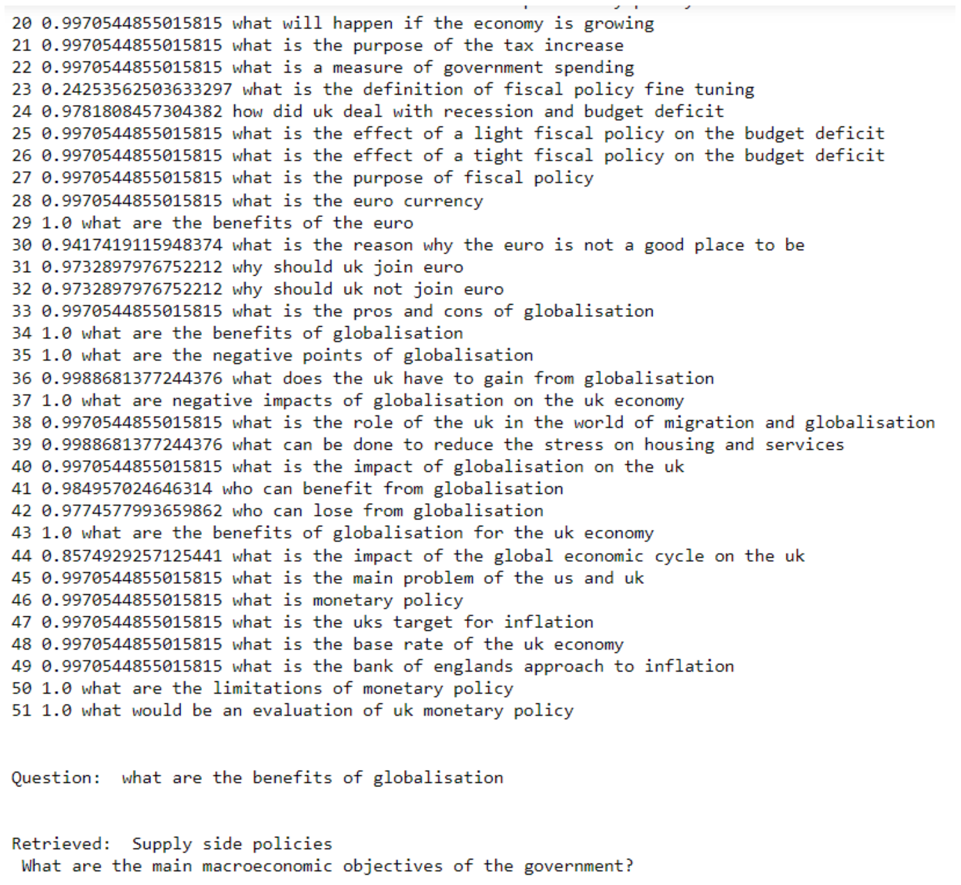
Results: This technique is implementation of the basic Natural Language Processing techniques and is quite powerful for basic text processing where documents are very different to one another.

Challenges: This process of handling and processing the text is age old and the industry as moved on from it. Also, there is no Natural Language Understanding.

5.2 Experiment II (Cosine Similarity and Sample Question Answering)

PART 1

Results: Here we see that using Basic Bag of Words NLP technique to make the corpus, but the similarity was not unique and incorrect. Hence the embeddings were not representing semantic information, refer Figure 38.



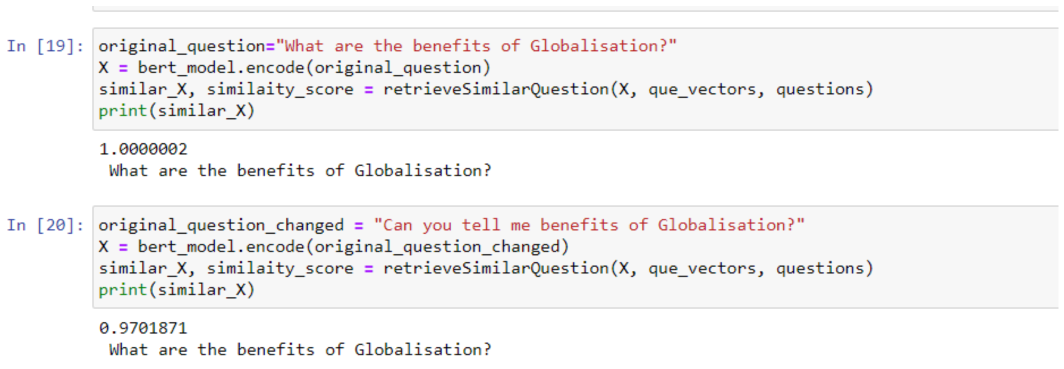
```
20 0.9970544855015815 what will happen if the economy is growing
21 0.9970544855015815 what is the purpose of the tax increase
22 0.9970544855015815 what is a measure of government spending
23 0.24253562503633297 what is the definition of fiscal policy fine tuning
24 0.9781808457304382 how did uk deal with recession and budget deficit
25 0.9970544855015815 what is the effect of a light fiscal policy on the budget deficit
26 0.9970544855015815 what is the effect of a tight fiscal policy on the budget deficit
27 0.9970544855015815 what is the purpose of fiscal policy
28 0.9970544855015815 what is the euro currency
29 1.0 what are the benefits of the euro
30 0.9417419115948374 what is the reason why the euro is not a good place to be
31 0.9732897976752212 why should uk join euro
32 0.9732897976752212 why should uk not join euro
33 0.9970544855015815 what is the pros and cons of globalisation
34 1.0 what are the benefits of globalisation
35 1.0 what are the negative points of globalisation
36 0.9988681377244376 what does the uk have to gain from globalisation
37 1.0 what are negative impacts of globalisation on the uk economy
38 0.9970544855015815 what is the role of the uk in the world of migration and globalisation
39 0.9988681377244376 what can be done to reduce the stress on housing and services
40 0.9970544855015815 what is the impact of globalisation on the uk
41 0.984957024646314 who can benefit from globalisation
42 0.9774577993659862 who can lose from globalisation
43 1.0 what are the benefits of globalisation for the uk economy
44 0.8574929257125441 what is the impact of the global economic cycle on the uk
45 0.9970544855015815 what is the main problem of the us and uk
46 0.9970544855015815 what is monetary policy
47 0.9970544855015815 what is the uks target for inflation
48 0.9970544855015815 what is the base rate of the uk economy
49 0.9970544855015815 what is the bank of englands approach to inflation
50 1.0 what are the limitations of monetary policy
51 1.0 what would be an evaluation of uk monetary policy


Question:  what are the benefits of globalisation


Retrieved:  Supply side policies
 What are the main macroeconomic objectives of the government?
```

Figure 38 Results of Experiment II, Part 1

Challenges: Again, age old technology of Bag of Words, Stemming and tokenization without semantic meaning

PART 2

Results: Here we have successfully fixed the problem of 'what is user does not ask the exact same question but a question that means the same as one in our knowledge base' using the BERT embeddings which preserves the semantic information. This is carried forward to the Experiment VI where it works beautifully.



```
In [19]: original_question="What are the benefits of Globalisation?"
         X = bert_model.encode(original_question)
         similar_X, similaity_score = retrieveSimilarQuestion(X, que_vectors, questions)
         print(similar_X)

         1.0000002
          What are the benefits of Globalisation?

In [20]: original_question_changed = "Can you tell me benefits of Globalisation?"
         X = bert_model.encode(original_question_changed)
         similar_X, similaity_score = retrieveSimilarQuestion(X, que_vectors, questions)
         print(similar_X)

         0.9701871
          What are the benefits of Globalisation?
```

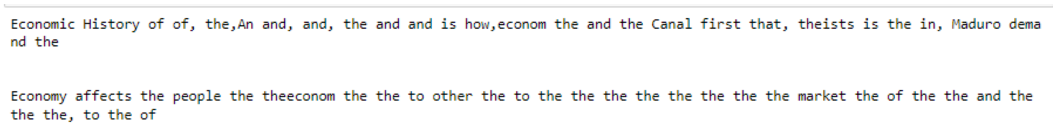
Figure 39 Experiment II Part 2 Comparison of Bag of Words and BERT embedding proving later as superior

Challenges: Now the user query similarity issue is resolved but a standalone model was not able to achieve the research objectives we had set in building an Opinion forming conversational AI.

5.3 Experiment III (Text Generation)

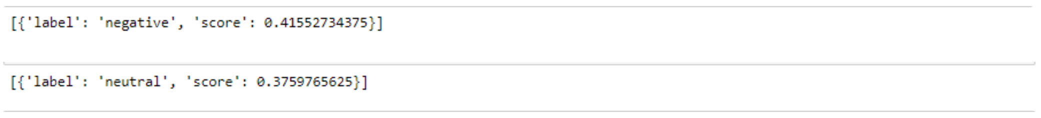
Results: We see from the output in Figure 40 that the output text sequence generated is gibberish and does not make sense, even after many epochs of training. But the sentiment analysis of the generated text has good results despite gibberish input in Figure 41. This method is then again extended to Experiment VI.

Challenges: Text generation failure happens due to the smaller number of training samples in dataset. This approach cannot be used for topic specific small size datasets.



```
Economic History of of, the,An and, and, the and and is how,econom the and the Canal first that, theists is the in, Maduro dema
nd the


Economy affects the people the theeconom the the to other the to the the the the the the the the market the of the the and the
the the, to the of
```

Figure 40 Experiment III Sequence Generated from fine-tuned GPT-2 model



```
[{'label': 'negative', 'score': 0.41552734375}]

[{'label': 'neutral', 'score': 0.3759765625}]
```

Figure 41 Experiment III Sentiment Analysis of generated text

5.4 Experiment IV (Transformer based text classification)

Results: The results obtained are very good at around 96.39% as viewed from the confusion matrix generated in Figure 42.

Challenges: This method is useful to know how to custom train a classification model with transformers since sentiment analysis is also a sequence classification problem.

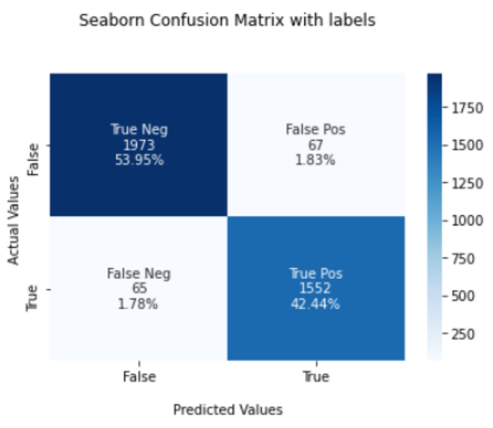
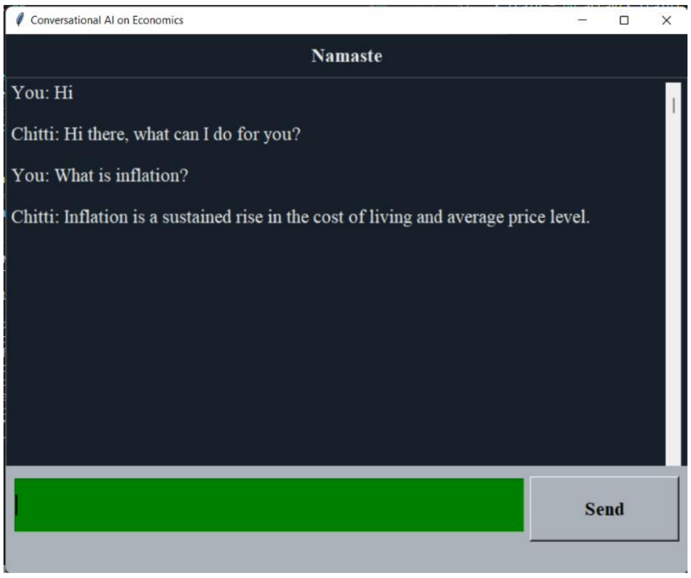


Figure 42 Experiment IV results via Confusion Matrix

5.5 Experiment V (Chatbot with Basic NLP)

Outcomes: Same problems as of experiments I and II(Part 1). Also, very limited or no Blended Skill Talk implementation, as it is increasingly difficult to code it into this AI. This kind of AI is good enough for web, e-commerce based chatbots. This is achieved using Standalone Neural Network.

Challenges: Failure to understand user context with varied inputs and opinion forming capability is absent. Also, sometimes out of context answer is printed this makes this AI unreliable. Refer to Figures 43,44.



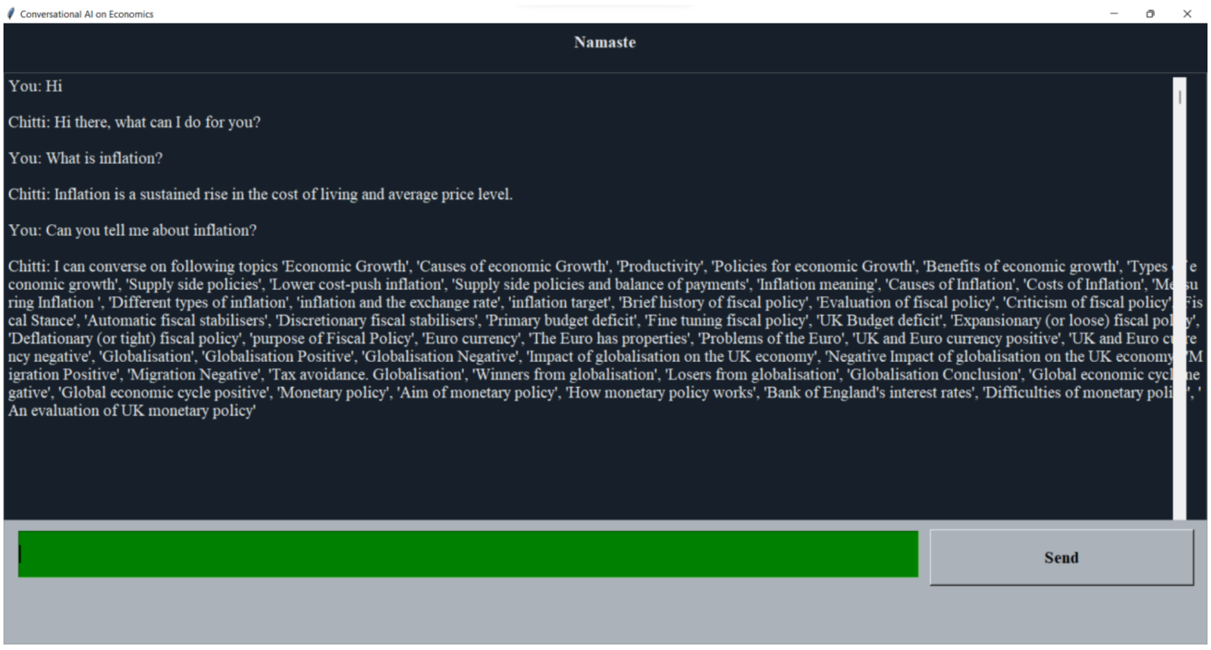
You: Hi

Chitti: Hi there, what can I do for you?

You: What is inflation?

Chitti: Inflation is a sustained rise in the cost of living and average price level.

Figure 43 Experiment V Chatbot Results



You: Hi

Chitti: Hi there, what can I do for you?

You: What is inflation?

Chitti: Inflation is a sustained rise in the cost of living and average price level.

You: Can you tell me about inflation?

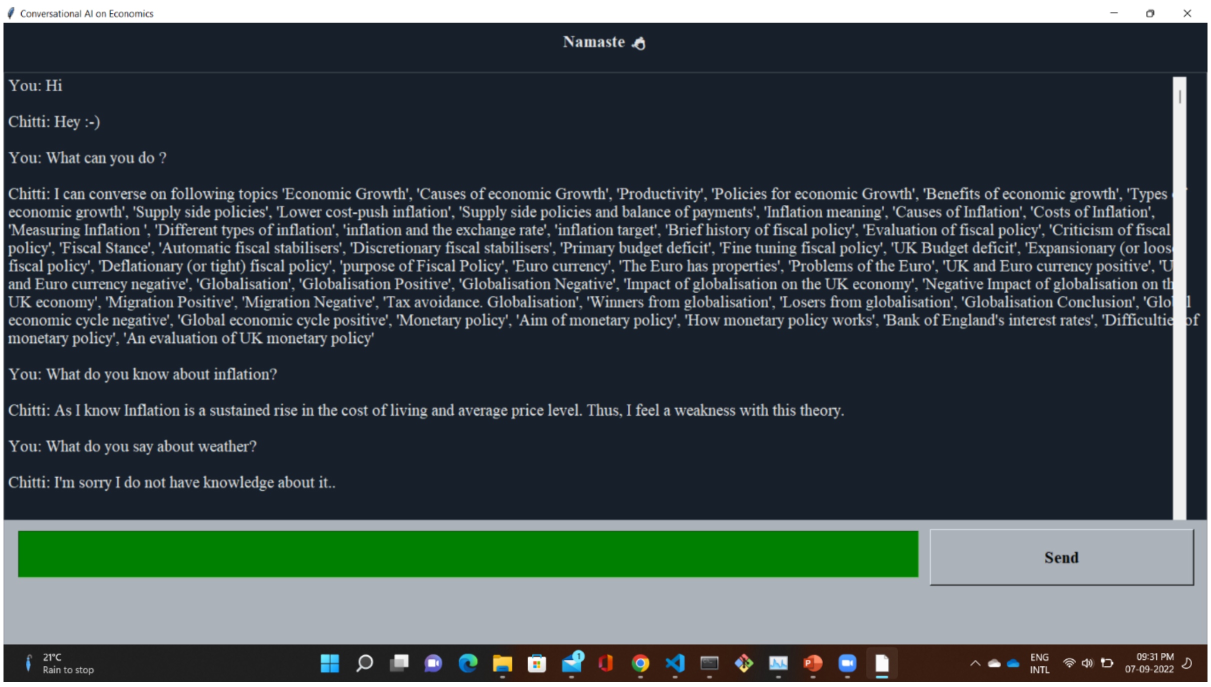
Chitti: I can converse on following topics 'Economic Growth', 'Causes of economic Growth', 'Productivity', 'Policies for economic Growth', 'Benefits of economic growth', 'Types e conomic growth', 'Supply side policies', 'Lower cost-push inflation', 'Supply side policies and balance of payments', 'Inflation meaning', 'Causes of Inflation', 'Costs of Inflation', 'Me su ring Inflation ', 'Different types of inflation', 'inflation and the exchange rate', 'inflation target', 'Brief history of fiscal policy', 'Evaluation of fiscal policy', 'Criticism of fiscal policy' Fis cal Stance', 'Automatic fiscal stabilisers', 'Discretionary fiscal stabilisers', 'Primary budget deficit', 'Fine tuning fiscal policy', 'UK Budget deficit', 'Expansionary (or loose) fiscal pol y', 'Deflationary (or tight) fiscal policy', 'purpose of Fiscal Policy', 'Euro currency', 'The Euro has properties', 'Problems of the Euro', 'UK and Euro currency positive', 'UK and Euro c re ncy negative', 'Globalisation', 'Globalisation Positive', 'Globalisation Negative', 'Impact of globalisation on the UK economy', 'Negative Impact of globalisation on the UK economy 'M igration Positive', 'Migration Negative', 'Tax avoidance. Globalisation', 'Winners from globalisation', 'Losers from globalisation', 'Globalisation Conclusion', 'Global economic cycl ne gative', 'Global economic cycle positive', 'Monetary policy', 'Aim of monetary policy', 'How monetary policy works', 'Bank of England's interest rates', 'Difficulties of monetary poli ', ' An evaluation of UK monetary policy'

Figure 44 Experiment V Chat Bot failure in conversation

5.6 Experiment VI (Opinion Forming Chatbot)

Results: Multi-modal based system using three Neural Networks sequentially. The conversational AI can understand user context with varied forms of inputs. It can give expected opinions based on knowledge database. Blended skill Talk is implemented and is completely working as shown in Figure 45.

Challenges: AI is unable to understand when wrong spelling is provided.



You: Hi

Chitti: Hey :-)

You: What can you do ?

Chitti: I can converse on following topics 'Economic Growth', 'Causes of economic Growth', 'Productivity', 'Policies for economic Growth', 'Benefits of economic growth', 'Types economic growth', 'Supply side policies', 'Lower cost-push inflation', 'Supply side policies and balance of payments', 'Inflation meaning', 'Causes of Inflation', 'Costs of Inflation', 'Measuring Inflation ', 'Different types of inflation', 'inflation and the exchange rate', 'inflation target', 'Brief history of fiscal policy', 'Evaluation of fiscal policy', 'Criticism of fiscal policy', 'Fiscal Stance', 'Automatic fiscal stabilisers', 'Discretionary fiscal stabilisers', 'Primary budget deficit', 'Fine tuning fiscal policy', 'UK Budget deficit', 'Expansionary (or loos fiscal policy', 'Deflationary (or tight) fiscal policy', 'purpose of Fiscal Policy', 'Euro currency', 'The Euro has properties', 'Problems of the Euro', 'UK and Euro currency positive', 'U and Euro currency negative', 'Globalisation', 'Globalisation Positive', 'Globalisation Negative', 'Impact of globalisation on the UK economy', 'Negative Impact of globalisation on th UK economy', 'Migration Positive', 'Migration Negative', 'Tax avoidance. Globalisation', 'Winners from globalisation', 'Losers from globalisation', 'Globalisation Conclusion', 'Glo l economic cycle negative', 'Global economic cycle positive', 'Monetary policy', 'Aim of monetary policy', 'How monetary policy works', 'Bank of England's interest rates', 'Difficultie of monetary policy', 'An evaluation of UK monetary policy'

You: What do you know about inflation?

Chitti: As I know Inflation is a sustained rise in the cost of living and average price level. Thus, I feel a weakness with this theory.

You: What do you say about weather?

Chitti: I'm sorry I do not have knowledge about it..

Figure 45 Opinion Generating Conversation AI shown at the time of Project Demo

5.7 Evaluation of Conversational AI

For this research paper since the dataset format created is different that the datasets used for evaluation of other Conversational AI's[14, 21, 23, 25, 26, 28, 29], we cannot evaluate using the current methodologies.

Since there exists no methodology or metric for evaluation currently, researchers use the gold standard of evaluation i.e., human evaluation. This makes perfect sense since these systems are to be ultimately used by humans and have a very big factor of HCI associated with them. So, we in this project will also use gold standard for evaluation.

In gold standard the evaluation criteria are that if the chat bot answers satisfactorily according to the user's perspective and set rules, then we considered it as a successful trial, else unsuccessful. Each correct expected response is given 1 point and incorrect ones are set to 0. There will be 5 user queries asked in each trial and we had 5 such trials.

After evaluation of the final chatbot application for 5 times here were the results:

Trial 1: Successful=5/5, Unsuccessful=0/5 Success Rate=100%

Trial 2: Successful=4/5, Unsuccessful=1/5 Success Rate=80%

Trial 3: Successful=5/5, Unsuccessful=0/5 Success Rate=100%

Trial 4: Successful=3/5, Unsuccessful=2/5 Success Rate=60%

Trial 5: Successful=5/5, Unsuccessful=0/5 Success Rate=100%

Chapter VI: CONCLUSION

Conversational AIs has always been an interesting topic among researchers especially Computer Linguists. However, creation of AI's that can generate their own opinions is a hot field of research currently. To generate opinion on the topic given according to its reasoning is a tough system to build, due to the enormous amounts of data and the format of data needed needs to extremely carefully handcrafted. This research has been successful in creating a conversational agent which has opinion forming capabilities and has achieved the objectives set by us for it.

6.1 Applications

- These types of AIs enable people to have human like conversation and get opinion of AI based on facts. Also, it expands accessibility for wider community, while reducing effort and resolution time.
- Enhances Human Computer interaction, and a lot of research is being done to make human friendly AIs. This project is a small step in that direction.
- By Building opinion capable AI, helps us to support and enhance research on understanding how opinion are formed by humans given a limited knowledge base on specific topic.

6.2 Future Works and directions

There is still a long way to go for opinion generation capable AIs to get to an acceptable standard. This research has successfully made a small step in that direction. But still there are some future works which can be done to enhance this AI, these will be stated and discussed as:

- Wrong Spelling correction. The current AI cannot work if there is a spelling mistake by the user in conversation. This is a very common human error and due to limited time of the project period this task is considered as a future improvement.
- Update knowledge database from user inputs on unknown knowledge automatically. Currently if the AI encounters an unknown question, for which it has no knowledge it simply returns a predefined statement. But this can be modified to create a new database of newfound knowledge from the user and add it to its database. After that it can expand its knowledge base by retraining again and be able to provide new information during next chat session.

REFERENCES

[1] [Internet]. 2022 [cited 20 September 2022]. Available from: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[2] Stemming vs Lemmatization in NLP: Must-Know Differences [Internet]. Analytics Vidhya. 2022 [cited 20 September 2022]. Available from: https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/

[3] Jiawei Han, Micheline Kamber, Jian Pei, 2 - Getting to Know Your Data, Data Mining (Third Edition), Morgan Kaufmann, 2012, Pages 39-82, ISBN 9780123814791, https://doi.org/10.1016/B978-0-12-381479-1.00002-2. (https://www.sciencedirect.com/science/article/pii/B9780123814791000022)

[4] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, Nils Reimers, Iryna Gurevych, arXiv:1908.10084v1 [cs.CL] 27 Aug 2019

[5] Back Propagation, Week 4, Neural Computation, Yunwen Lei, University of Birmingham, 2021

[6] [Internet]. 2022 [cited 20 September 2022]. Available from: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

[7] Gradient descent (article) | Khan Academy [Internet]. Khan Academy. 2022 [cited 20 September 2022]. Available from: https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent#:~:text=Gradient%20descent%20is%20an%20algorithm,like%20we've%20seen%20before.

[8] Attention is All you Need, Vaswani, Parmar, Jones, arXiv:1706.03762v5 [cs.CL] 6 Dec 2017

[9] [Internet]. 2022 [cited 20 September 2022]. Available from: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[10] [Internet]. 2022 [cited 20 September 2022]. Available from: https://www.topbots.com/transfer-learning-in-nlp/

[11] Text Summarization with NLP: TextRank vs Seq2Seq vs BART [Internet]. Medium. 2022 [cited 20 September 2022]. Available from: https://towardsdatascience.com/text-summarization-with-nlp-textrank-vs-seq2seq-vs-bart-474943efeb09

[12] The basics of NLP and real time sentiment analysis with open source tools [Internet]. Medium. 2022 [cited 20 September 2022]. Available from: https://towardsdatascience.com/real-time-sentiment-analysis-on-social-media-with-open-source-tools-f864ca239afe

[13] facebook/bart-large-cnn · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022]. Available from: https://huggingface.co/facebook/bart-large-cnn

[14] Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills, Eric Smith, Jason Weston, Kurt Shuster, arXiv:2004.08449v1 [cs.CL] 17 Apr 2020

[15] Python (programming language) - Wikipedia [Internet]. En.wikipedia.org. 2022 [cited 20 September 2022]. Available from: https://en.wikipedia.org/wiki/Python_(programming_language)

[16] NLTK :: Natural Language Toolkit [Internet]. Nltk.org. 2022 [cited 20 September 2022]. Available from: https://www.nltk.org/

[17] scikit-learn - Wikipedia [Internet]. En.wikipedia.org. 2022 [cited 20 September 2022]. Available from: https://en.wikipedia.org/wiki/Scikit-learn

[18] scikit-learn: machine learning in Python — scikit-learn 1.1.2 documentation [Internet]. Scikit-learn.org. 2022 [cited 20 September 2022]. Available from: https://scikit-learn.org/stable/

[19] PyTorch [Internet]. SearchEnterpriseAI. 2022 [cited 20 September 2022]. Available from: https://www.techtarget.com/searchenterpriseai/definition/PyTorch

[20] TensorFlow - Wikipedia [Internet]. En.wikipedia.org. 2022 [cited 20 September 2022]. Available from: https://en.wikipedia.org/wiki/TensorFlow

[21] Neural Network-Based Abstract Generation for Opinions and Arguments, Lu Wang, Wang Ling, 9 Jun 2016, arXiv:1606.02785v1

[22] Information Propagation and Public Opinion Evolution Model Based on Artificial Neural Network in Online Social Network, Xiaoyang Liu, Daobing He, November 2020, Pages 1689–1703, https://doi.org/10.1093/comjnl/bxz104

[23] Scaling Language Models: Methods, Analysis & Insights from Training Gopher, DeepMind, arXiv:2112.11446v2 [cs.CL] 21 Jan 2022

[24] Wagner, D.N. Economic patterns in a world with artificial intelligence. Evolut Inst Econ Rev 17, 111–131 (2020). https://doi.org/10.1007/s40844-019-00157-x

[25] Unsupervised Opinion Summarization as Copycat-Review Generation (Bražinskas et al., ACL 2020)

[26] Automatic Opinion Question Generation (Chali & Baghaee, 2018)

[27] Learning Sentence Ordering for Opinion Generation of Debate (Yanase et al., 2015)

[28] Neural Network-Based Abstract Generation for Opinions and Arguments (Wang & Ling, NAACL 2016)

[29] Self-Supervised Multimodal Opinion Summarization Im et al., ACL 2021 https://aclanthology.org/2021.acl-long.33

[30] 🤗 What's Hugging Face? [Internet]. Medium. 2022 [cited 20 September 2022]. Available from: https://towardsdatascience.com/whats-hugging-face-122f4e7eb11a

[31] Sentence Transformers and Embeddings | Pinecone [Internet]. Pinecone. 2022 [cited 20 September 2022]. Available from: https://www.pinecone.io/learn/sentence-embeddings/

[32] sshleifer/distilbart-cnn-12-6 · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022]. Available from: https://huggingface.co/sshleifer/distilbart-cnn-12-6

[33] siebert/sentiment-roberta-large-english · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022]. Available from: https://huggingface.co/siebert/sentiment-roberta-large-english

[34] ProsusAI/finbert · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022]. Available from: https://huggingface.co/ProsusAI/finbert

[35] gpt2 · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022]. Available from: https://huggingface.co/gpt2

[36] Economics For Dummies Cheat Sheet - dummies [Internet]. Dummies.com. 2022 [cited 20 September 2022]. Available from: https://www.dummies.com/article/business-careers-money/business/economics/economics-for-dummies-cheat-sheet-208692/

[37] Contextual Chatbots with Tensorflow [Internet]. Medium. 2022 [cited 20 September 2022]. Available from: https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077

[38] GitHub - python-engineer/pytorch-chatbot: Simple chatbot implementation with PyTorch. [Internet]. GitHub. 2022 [cited 20 September 2022]. Available from: https://github.com/python-engineer/pytorch-chatbot

[39] Pettinger T. Economics A-Z - Economics Help [Internet]. Economics Help. 2022 [cited 20 September 2022]. Available from: https://www.economicshelp.org/economics-a-z/

[40] Fake News | Kaggle [Internet]. Kaggle.com. 2022 [cited 20 September 2022]. Available from: https://www.kaggle.com/c/fake-news/data

[41] distilbert-base-uncased-finetuned-sst-2-english · Hugging Face [Internet]. Huggingface.co. 2022 [cited 20 September 2022].

Appendix

Git Repository provided by University of Birmingham:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2021/rrp187/-/tree/master/

Structure:

The repository is structured in the format of the Experiment numbers as mentioned in Section 4 under the folder "Final Code for Submission". The final code to run is in the Experiment 6 folder.
There is also another folder by the name "Web Scrapping and Datasets" which has the initial datasets. Proper datasets can be found in the corresponding Experiment number folder.

Executing Code:

Firstly Python 3.9 must be installed either as standalone or with anaconda and environment variables must be set correctly. We will need to install NumPy, sklearn, PyTorch, Tensorflow, Transformers, NLTK, Punket, Tkinter, sentence_transformers and json and any other dependencies if required.
Download or open the folder Experiment 6 in an editor, VS code is recommended and set python interpreter path correctly.

Run in terminal "python bot_app.py"