Keyword Summarizer Using BERT

Aditya Goutam
School of Computer Engineering and Technology
MIT Academy of Engineering
Pune, Maharashtra
Email: aditya.goutam@mitaoe.ac.in

Prabhu S Mane
School of Electrical Engineering
MIT Academy of Engineering
Pune, Maharashtra
Email: psmane@mitaoe.ac.in

Nehaal Pandey
School of Computer Engineering and Technology
MIT Academy of Engineering
Pune, Maharashtra
Email: nehaal.pandey@mitaoe.ac.in

Gauri Gupta
School of Computer Engineering and Technology
MIT Academy of Engineering
Pune, Maharashtra
Email: gauri.gupta@mitaoe.ac.in

Chetan Patil
School of Computer Engineering and Technology
MIT Academy of Engineering
Pune, Maharashtra
Email: chetans.patil@mitaoe.ac.in

Rudragouda G Patil
Assis. Prof., School of Computer Engineering and Technology
MIT Academy of Engineering
Pune, Maharashtra
Email: rgpatil@mitaoe.ac.in

Abstract— The Data generated everyday is exponentially rising. Hence, a need arises for information extraction and utilisation of the same. Automatic Text Summarization is one of the crucial aids offered by Natural Language Processing (NLP). The problem with embedding models is it cannot process the words for which it is not trained. Hence, embedding models fail to deliver expected text synthesis.

This paper proposes a deployable keyword summarizer using BERT. BERT is a pre-trained transformer model that follows masked language modelling. Hence, it can effectively handle large amounts of data and extract necessary information for efficient utilisation and data synthesis.

Keywords: Automatic Text Summarization, Information Extraction, Natural Language Processing, BERT model.

I. Introduction

The 21st century is called the 'century of data'. Professionals across different walks of life identify data as the new oil. In 2022, the per day Data generation stats stand at a staggering 2.5 quintillion bytes, the unit conversion of which to Gigabytes becomes a head-spinning task. When we only look at the text messages generated per day across the globe, the numbers range somewhere near 18 Billion messages. Apart from text messages, text generated through documents, articles, guides, Business reports etc. contribute significantly to the numbers. With the technological advancements, these numbers would continue to inflate with time. Thus, with such tremendous data generation, we can say a lot of information is created everyday. Information extraction and utilisation is one of the major focuses for research in the current milieu. A lot of data generated contributes to the digital debris as they contain unnecessary or irrelevant content. Thus, researchers across the globe focus majorly on text summarisation for information extraction.

Manual Text summarisation is one of the effective ways for generating synopsis for a text. It is time consuming, computationally exhaustive and requires complete manual labour. Hence, for large text occurring as teradata, this is not a feasible option. With the advent of Artificial intelligence and Machine Learning, a significant development has been made in using Natural Language Processing (NLP) for text analysis and synthesis. Using NLP, a lot of effort has been made to build an Automatic Text Summarisation tool through different methods. The ATS tool focuses on consuming large amounts of text as input and processing it to make it as condensed as possible, encapsulating as much information as possible. This helps in clearing a lot of clutter around the information, thus extracting only the relevant parts from the text. Earliest References of Text summarisation dates back to the 1950s, developed by Luhn, which was also used in the first commercial computer, the IBM 701. He used the concept of bag-of-words for Text summarisation. Since then, we have seen a lot of development and new methods like word2vec, neural word embedding along with deep learning approaches by using Recurrent Neural Networks (RNN), concisely Long Short Term Memory(LSTM) networks.

The major drawback of embedding models is that it cannot predict and handle the words which are out of the vocabulary since it identifies a word's context in the document before generating the vocabulary.

We propose BERT which is Bidirectional Encoder Representations from Transformers which would solve the issue of handling the out of vocabulary words as it is pre trained using the English Wikipedia and the Brown Corpus. BERT incorporates Masked Language Modelling, meaning it masks the words and then predicts the masked words. Based on the parameters BERT is divided into two parts namely BERT BASE and BERT LARGE.

The main contribution of this paper is to deploy a fully working keyword summarizer using the BERT model. The input will be fetched from the user as a keyword of which he will receive a generated summary related to the input keyword. Backend consists of a procedure for scrapping information about the keyword and enforcing a transformer for effective summarization.

II. REVIEW OF LITERATURE

Salam et.al [1] produced a study on the effectiveness of BERT-based model variants for text summarization through diverse experimentations. Paper proposes "SqueezeBertSum" which is a trained summarization model finely tuned with the SqueezeBERT encoder variant. Paper successfully obtained competitive ROUGE scores while retaining the BERTSum baseline model performance of 98%. SqueezeBERT produces the same ROUGE-1 value as DistilBERT but having about 20% fewer parameters. It also produces marginally better ROUGE-2 and ROUGE-L values.

Hanumanthappa et.al[2] compares various techniques of Semantic analysis using Natural Languages Processing. The two important Research fields acknowledged here are the LSA model and Ontology. LSA is used for extracting data and deriving information from text that has semantic content. Using the ontology technique, structure data is extracted from unstructured data, information is derived from databases, and applications on the semantic web.

Wang et.al [3] present a novel extractive-abstractive hybrid model that employs BERT on words embedded with reinforcement learning. RL is used to bridge the extractive network with the abstractive network. Using the CNN/Daily Mail dataset and the ROUGE metrics as the assessment method, the model was assessed with one of the most widely used automatic text summary models in order to confirm its effectiveness. Results from extensive experiments reveal that the model's accuracy has improved significantly.

Chen et.al [4] suggest a solution to the problem of the BERT model overfitting on smaller datasets. Overfittinglowers the model's performance to a great extent. Self-Supervised Attention, a novel method for enhancing and incorporating the BERT model, is suggested by the author. This hybrid model exhibits a notable performance enhancement.

Devlin et.al[5] introduced a new representational language model called BERT (Bidirectional Encoder Representations from Transformer) which would improve the fine tuning based approach for language representation. There are two approaches for applying pre-trained language representations namely feature based and fine tuning. Both of these approaches have the same objective but for the pre trained approaches there is a restriction for fine tuned approaches. These are unidirectional and would either use a left to right architecture or right to left architecture. BERT is a Masked Language Model which would overcome the constraint of a unidirectional approach by masking a random token and then predicting it.

Vaswani et.al [6] have tried to emphasise the importance of self attention in a transformer. Transformer consists of encoder and decoders which consist of feed forward neural networks and self attention layer. The attention layer lets us know the extent of preference to be given to a particular word. To determine this preference, there is a scaled dot product attention formula which is derived using 5 steps. If many words are to be considered then the paper defines the concept of multi headed attention.

Miller et.al [7] has an approach to use BERT along with K Means clustering for the extractive text summarization on a whole lecture. It compares the previously built methods which ran manually and projects such as M.I.T.'s lecture processing. This paper firstly tokenizes the words which are then passed to BERT for output embeddings, after which the embeddings are clustered using K Means and then selecting the embeddings that are closest to the centroid.

Idris et.al [8] draws a Study on the "Efficient Way of Web Development Using Python and Flask". Starting with an Introduction to Python and Flask, Python is a High-Level Programming language that helps in developing Dynamic web applications with efficient structure and elegant design. Python with the advantage of Clear and Orthogonal Syntax, Easy support for default arguments, and Object-Oriented file handling makes it easy for development.The paper further introduces Flask which is a micro framework written in Python on the WSGI toolkit and Jinja2 template engine, that provides different libraries and sets of tools for the development of a web application, it has two parts static file and template files. The paper also contains code-written examples of Flask. The Paper also helps us in understanding Template Inheritance and File Organisation. In Template Inheritance, the base template is overridden by the child template by using %block% and %extend%, whereas File Organisation is a way in which the files are arranged in the directory.

Further, OpenCV which is a Library of Python that deals with computer vision is introduced. The main application of this library is in recognition systems, object identifications, and tracking. Eventually, a Hand Gesture project was explained, which can be implemented with the help of OpenCV, Python, and Flask. The flow for the same starts with segmenting the hand region, Background subtraction, Motion detection and thresholding, Contour Extraction, Counting fingers. The results for the same end with counting the exact number of fingers present in the frame.

Lunn et.al [9] demonstrates how web-scraping can be useful in extracting data from publicly available web pages and in addition they also discuss how natural language processing can be useful to obtain salient information from textual data. Also, they have demonstrated techniques that can be used for numerous applications to further knowledge in CSE. Also, they show how the application of web-scraping and NLP are useful in obtaining and analysing pertinent information from the internet.

III. METHODOLOGY

The principal outcome is to provide a deployed project that will consume an input keyword inputted by the user on the web page, pass it on to the data mining phase; providing a web scrapped paragraph related to the keyword, and then returning the crude paragraphs to the machine learning orchestration, which will return the summarised data to back to the user. This approach has been implemented and divided into three phases.

Phase 1: Front End Development

Our first phase primarily consists of creating a front end for accepting the input from the user and then displaying the output after our input has been modelled. Further on, architecture has been deployed on AWS (Amazon web services) cloud's EC2(Elastic Cloud Compute) virtual server. For the creation of the frontend interface Html rendering CSS has been incorporated. The flask library for web development needs to be connected in a local manner. For this, the flask library needs to be installed onto local environments which will work in the backend and is capable of controlling and handling all the data processing required to support a full-featured frontend. Model has been deployed on AWS using an EC2 server with Ubuntu Operating System. After adding storage, security groups for handling inbound and outgoing traffic, nginx, we can Review and launch our web page.

Phase 2: Machine Learning

The implementation of this phase started with installing and importing all the required modules and libraries which were in requirements.txt file. Some of the important ones are related to transformers and BERT for the text summarization part. Now the next step is to use the 'distilbert-base-uncased' model from the Summarizer. We have used distilbert as it is a lighter version of the main BERT BASE and is easy to run on any operating device. Once the model is imported then we give the input sentences or paragraphs as an input to the model along with the minimum number of sentences for getting the summarised output. The subroutine bounces back the summarised data to the front end. Machine Learning is segmented into three subsections:

a. Word Embeddings:

Word embedding is used for the representation of document vocabulary. It basically captures the context of a word present in the document which is done by vector representations of the occurring words. Parameters of the vector can take values for one hot encoding or could be initialised as their occurrences across the vocabulary. Embeddings signify word meaning in a spatial locality, meaning that synonymous words would be near to each other in the hyperspace. There are several models for evaluating word embeddings:

i) CBOW Model:

Model takes the context of each word present as an input and then tries to predict the word corresponding to the context. The input is a hot encoded vector of size V, the hidden layer contains N neurons and the output is again a length vector V but the elements being the softmax values. The hidden layer just copies the weighted sum of inputs to the next layer. Here there is no role activation like sigmoid and tanh. We only deal with the softmax calculation present in the output layer.

ii) Skip-Gram Model:

The target word is inputted to the network and then based on the provided input, the model gives us the output C probability distributions of V probabilities, one for each word.

a. Attention Mechanism

Attention is the extent of preference given to a word in the sentence. Attention is quantified by following the below steps in accord with fig. :

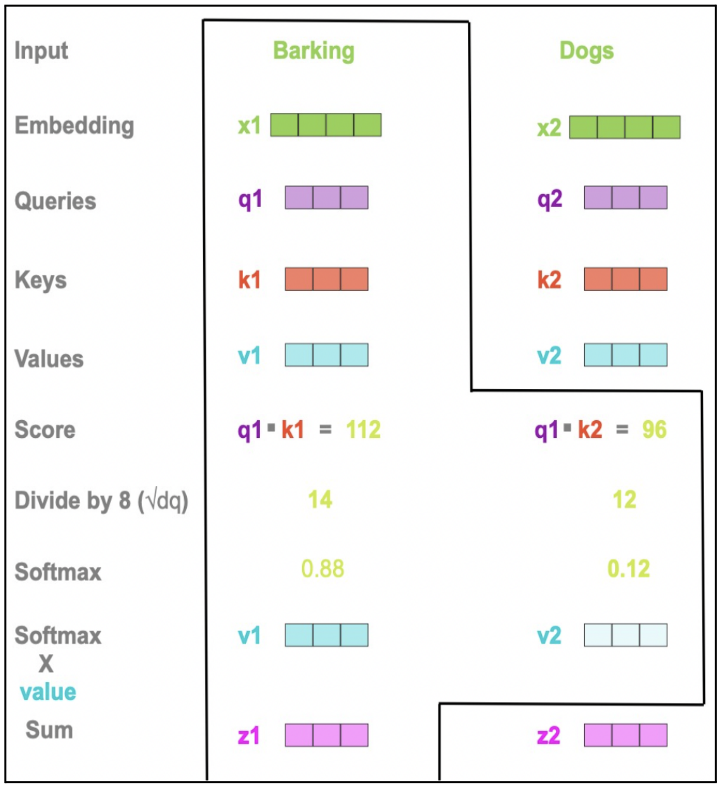

Fig.1 (Single Head Attention)

1. Three weights are taken and random values are assigned which would result in a vector having 64 dimensions and then backpropagation will take place in order to get the correct weights. The 3 weights Query, Keys and Values are multiplied by the input vectors and then we get particular values like q, k, v.

2. We have to multiply q1 with k1 and k2 i.e. with respect to word 1. Similarly multiplying with respect to word 2 we obtain the result that is known as score.

3.For word 1 we need to divide it by 8 (under root (dx), where dx is 64) with the value of score.

4. Applying the softmax which has a total value of 1. The need of softmax is to basically get the most important between the two which is purely based on probability.

5. Multiplying softmax with values we get the vectors. Suppose we get v1 and v2, so we take the sum of the two to get the final z1 which will then be passed to the feed forward neural network.

Underlying architecture in transformer learning comprises encoder and decoder stacked upon each other.

Above mechanism was first put forward in "Attention Is All You Need", to overcome complex intricacies of recurrent or convolutional neural networks. It was devised to improve the accuracy of English-to-German translation.

b. Transformer Learning:

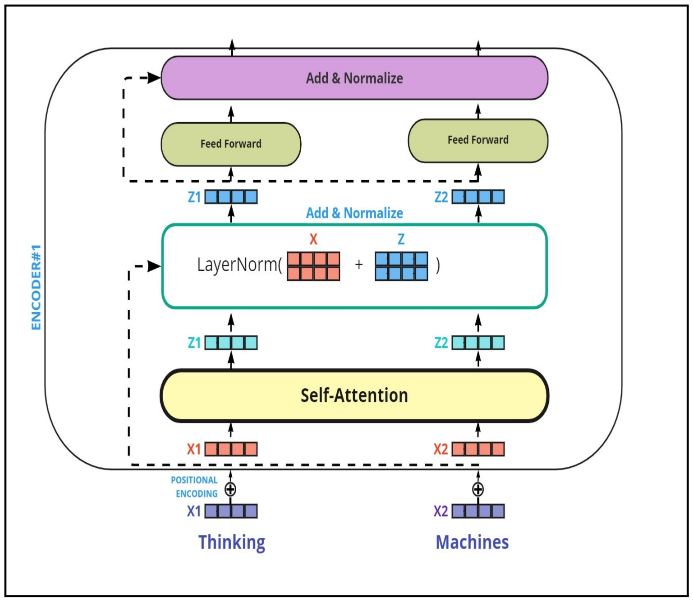

Fig.2 (Transformer Architecture)

The governing architecture of the transformer learning model comprises encoder and decoder as given in Fig. 2. Encoder consists of two layers: Self Attention and the Feed Forward Neural Network. The input to the encoder is converted to dimension vectors by using embedding which in this case is the Word2Vec (V=512). Vector is passed to the self attention layer at the same time i.e. parallel and not one at a time.

Decoder is same as an encoder only it has an extra layer of encoder-decoder attention in which the output of the encoder is given as an input to it. The generated output from the decoder after converting it into linear followed by softmax is given input to the decoder as parallel. This process is continued until EOS or End Of Statement.

Phase 3: Data Science

Web scraping is one of the essential tasks of semantic analysis, hence it is of utmost importance to extract the data from the web to complete the process. The complete process of obtaining the knowledge from the web can be seen in the fig 1.1.

Python package library Beautiful Soup is used to extract the data from the web. The extracted data then can be used for natural language processing and further mask language modelling by bert.

i.) Libraries:

The libraries- beautifulsoup and selenium can be used for web scraping. Both beautifulsoup and selenium have their pros and cons. Beautifulsoup works effectively when used for data extraction from a static website, but is ineffective when it comes to dynamic website, where in Selenium can be effectively used for data extraction from a dynamic website but requires skilled implementation. Hence selenium is used when data is to be extracted from a dynamic website and beautifulsoup is used when extracting data from a static website.

ii.) Web-Scraping and Data extraction:

Web scraping is widely used to extract data from the web as the extracted data can then be used for various purposes like customer review, semantic analysis, etc. Process of web-scraping is divided into three sections. In section A, we will discuss on Http request to the server, in section B we will discuss on Extracting and parsing data and in section C we will discuss on Storing the data.

A. Making an HTTP Request to the server:

Making a HTTP request to the server is the foremost step for extracting the data from the web. Whenever a browser makes an HTTP request to the server, it then sends the HTML source code of the webpage to the browser which is then converted by the browser into a beautiful webpage. Similarly, after making an HTTP request, the source code is obtained in the process of web-scraping which is then used for extracting data.

B. Extracting and Parsing Data:

Now, the obtained source code is inspected to find the required class, but before this we need to parse the source code as the source-code is in unstructured format. Parsing the html code means to convert the unstructured code into a structured code so that the tags and the data can be differentiated. So, python modules like lxml.html and beautifulsoup can be used to parse the html source-code. The lxml.html module is used when dealing with a complex html script but as we are dealing with a static html webpage so we are using beautifulsoup module to parse the data. After parsing the data, the specific class which is required is extracted from the code to obtain the necessary data.

The most ubiquitous data which internet sites usually collect include images, text, videos, product information, customer sentiments, reviews, and pricing from comparison websites. After obtaining the data, it can then be stored in various formats.

C. Saving the Relevant Data Locally:

After extracting the data from the source code, the data is to be stored somewhere so that it can be used by other processes. The obtained data can be stored in various formats like XML, XLSX, SQL, TXT, etc. Also, the data can be stored in a variable which then can be directly in the program for further processing.

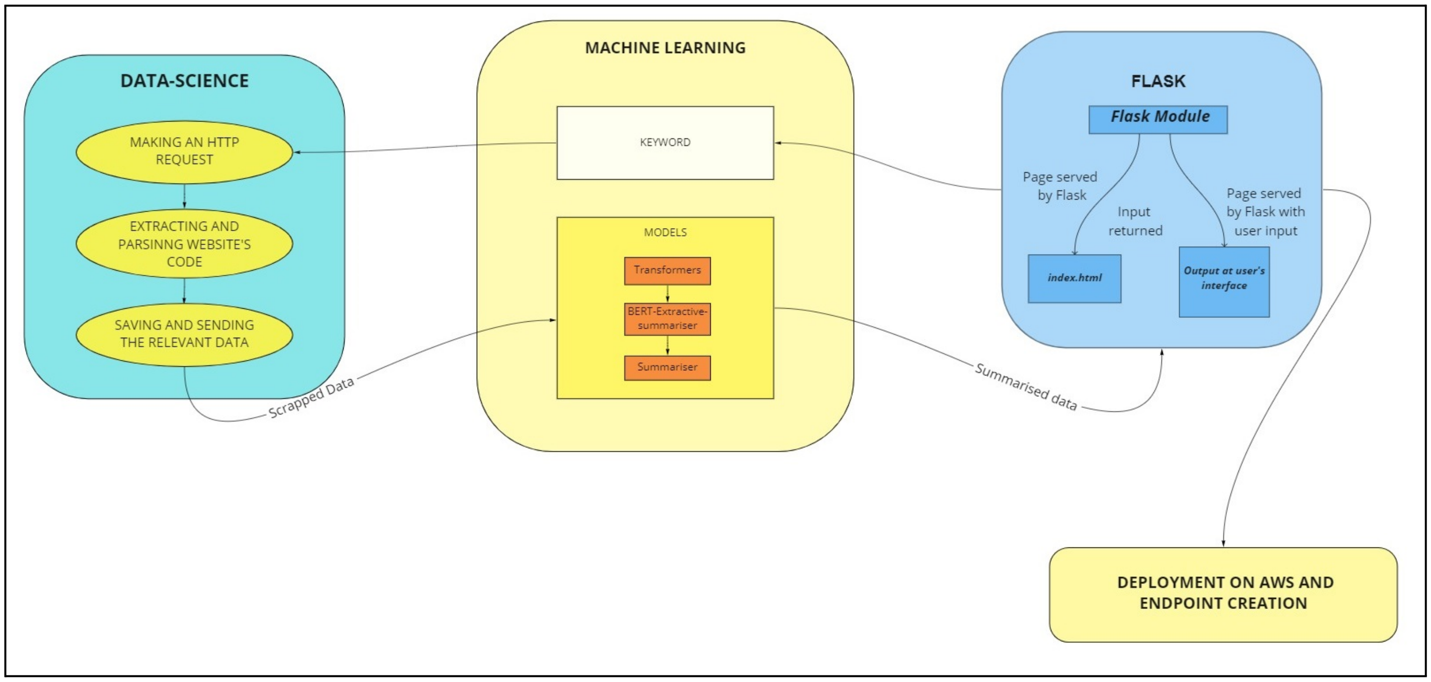
DATA-SCIENCE
MAKING AN HTTP REQUEST
EXTRACTING AND PARSINNG WEBSITE'S CODE
SAVING AND SENDING THE RELEVANT DATA
Scrapped Data
MACHINE LEARNING
KEYWORD
MODELS
Transformers
BERT-Extractive-summariser
Summariser
Summarised data
FLASK
Flask Module
Page served by Flask
Input returned
Page served by Flask with user input
index.html
Output at user's interface
DEPLOYMENT ON AWS AND ENDPOINT CREATION

Fig.3 (System Architecture)

VII. SYSTEM ARCHITECTURE

The architecture's main theoretical objective is to comprehend how our system truly operates. A semantic system is divided into three components, which are as follows: With precise user input, the global server searches the word or the phrase, then scrapes the most relevant and idealistic data from the website. This data is then modelled to our text summariser, BERT, which summarises the entire text and displays it at the user's end as shown in the Fig.3.

VII. CHALLENGES

1. Frontend:

i) Connecting the embedded flask code to the HTML code. It was displaying syntactical errors, but after some searching, we eventually passed it.

ii) Deploying the website and code to AWS cloud. Because of certain unknown configuration issues, this was tedious. We were able to overcome the problem by using the right instance, storage, and environment.

2. Machine Learning:

i) Installing the modules and libraries via requirements.txt required for the BERT model to run was not working on all operating systems as many modules were outdated and not working with the current versions of the other modules.

ii) BERT BASE is a huge model which is around 1.5 GB size so it would require much time to install it and use it again and again. Condensing the entire text is an error prone task, the summariser often neglects processing the first sentence.

3. Web-Scraping:

i) Taking into account search engine optimization and preferential indexing based on volume of incoming traffic is considerably arduous.

ii) Operating on datalakes occurring in teradata for augmenting semantic footprints is not scalable.

iii) Resultant web page script stored in soup object after web-scraping often occurs as HTML objects. This poses a challenge
for further text summarization.

iv) Dynamic Site: It's easy to scrape a static website as the structure of its code is the same for every page, but when it comes to dynamic web pages, the structure of code is different for different pages. Hence it becomes difficult to scrape such web pages.

VIII. RESULTS

I. Finalised Architecture:

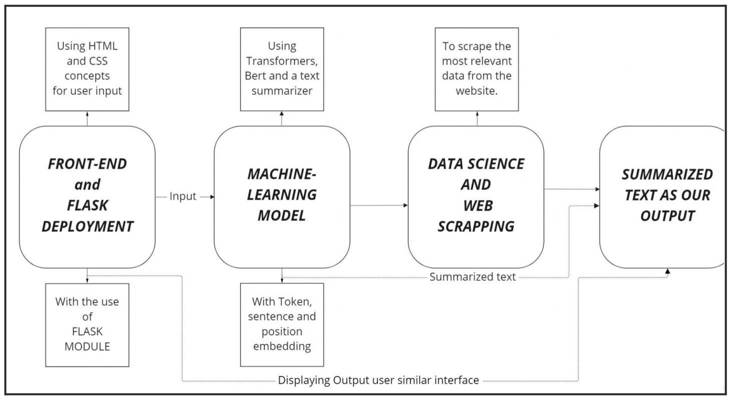
Using HTML and CSS concepts for user input
Using Transformers, Bert and a text summarizer
To scrape the most relevant data from the website.
FRONT-END and FLASK DEPLOYMENT
Input
MACHINE-LEARNING MODEL
DATA SCIENCE AND WEB SCRAPPING
SUMMARIZED TEXT AS OUR OUTPUT
Summarized text
With the use of FLASK MODULE
With Token, sentence and position embedding
Displaying Output user similar interface

Fig.4 (Finalised Architecture)

II. Trade-off comparison between BERT Architectures:

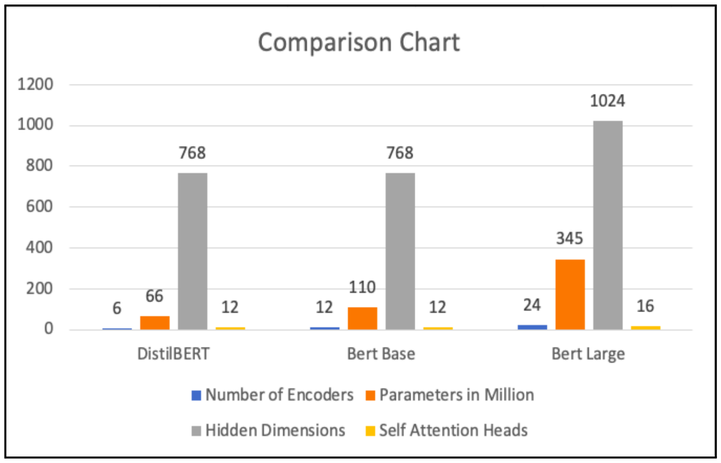

Fig.5 (BERT architecture comparison charts)

IX. CONCLUSION

The following paper presented a pipeline for text summarization using transformer learning using "distil-BERT" as an underlying architecture. Reducing verbosity in the age of data inflation and automation is one of the major objectives of the paper. Moving forward, we aim to incorporate semantic indexing and operations on data lakes. Making text summarisation feasible apart from extractive text analyses has been one of the key focuses of the paper.

X. ACKNOWLEDGEMENT

We would like to thank the School of Computer Engineering and Technology at MIT Academy of Engineering, Alandi for the opportunity to conduct this research. We also greatly appreciate the support provided by Prof. Krunal Pawar sir and the REDx AI Club, MITAOE for helping us present our research.

XI. REFERENCES

[1] Abdel-Salam, S.; Rafea, A. Performance Study on Extractive Text Summarization Using BERT Models. https://doi.org/10.3390/info13020067

[2] Rajani, M. Hanumanthappa, Techniques of Semantic Analysis for Natural Language Processing – A Detailed Survey. https://ijarcce.com/wp-content/uploads/2016/11/IJARCCE-ICRITCSA-32.pdf

[3] Qicai Wang, Peiyu Liu, Zhenfang Zhu, Hongxia Yin, Qiuyue Zhang and Lindong Zhang, 2019, https://www.researchgate.net/publication/337014455_A_Text_Abstraction_Summary_Model_Based_on_BERT_Word_Embedding_and_Reinforcement_Learning

[4] Y. Chen, X. Kou, J. Bai and Y. Tong, "Improving BERT With Self-Supervised Attention," in IEEE Access, vol. 9, pp. 144129-144139, 2021, doi: 10.1109/ACCESS.2021.3122273

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2019, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin, 2017, Attention Is All You Need, arXiv:1706.03762

[7] Derek Miller, 2017, Leveraging BERT for Extractive Text Summarization on Lectures arXiv:1906.04165

[8] Nuruldelmia Idris, Cik Feresa Mohd Foozy, Palaniappan Shamala, 2020, A Generic Review of Web Technology: DJango and Flask, ISSN 2714-7533

[9] Stephanie Lunn,Jia Zhu, Monique Ross ,Utilising Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice,https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9274270