NLP approach to summarize the business news about stocks and finding their sentiments

By

Shubham Sakunde

Student ID: 2361173

Supervisor: Dr. Mohammed Bahja

UNIVERSITY OF BIRMINGHAM

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

Table of Figures

Table of Tables

ABSTRACT

In the world, about 13% of people do trade them in stock market actively with consideration of different factors. The stock market works on human emotions or mass psychology. News related to stocks, or a particular sector is one consideration. Business news is the one that makes huge impact on stocks, or a particular sector whenever released. It contains different information such as the company's annual or monthly reports, information about the company's strengths and weaknesses, etc. The information in news are bit large and lengthy to understand in one go. It takes a lot of time to read the whole news in a minimum time. Also, nowadays some peoples trade blindly without knowing the impact of news on that stock or that sector.

In this project, we developed a system that will give us a summary of that news as well as gives its sentiment based on the heading of that news. Here we have used a multi-modal approach for two different tasks. For summarization, we retrained the T5 base model and fine-tuned that on our dataset as well as for sentiment analysis we used the BERT base model. To evaluate the model results here ROUGE score and F1 Score have been used for summarization and sentiment analysis respectively. In the final stage, we have created a web application interface that will give us summaries and sentiments of unknown news. We achieved results better than state-of-the-art models like we got average Rouge score as R1 0.58, R2 0.48, RL 0.56 and for sentiment analysis we got highest accuracy as 0.87.

Keywords: Natural language processing, T5, BERT, Summarization, Sentiment analysis, Rouge, F1 score

Acknowledgment

I would like to acknowledge and give my warmest thanks to my supervisor Dr. Mohammed Bahja who made this work possible. His guidance and advice carried me through all the stages of writing my project

I would also like to give special thanks to my parents and my family as a whole for their continuous support and understanding when undertaking my research and writing my project. Your prayer for me was what sustained me this far.

Finally, I would like to thank God, for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree. I will keep on trusting you for my future.

CHAPTER I: Overview

1.1: Introduction

In this chapter, we will see some introduction about our project, research question, project objectives, and the outline of the further chapters.

What is a summarization of business news?

Summarization is the process of producing a condensed version of a document while maintaining its key points. Some models are able to extract text from the original input, while others may generate completely new text. Especially, in news maintaining the key point in the news summary is the crucial task for the machine.

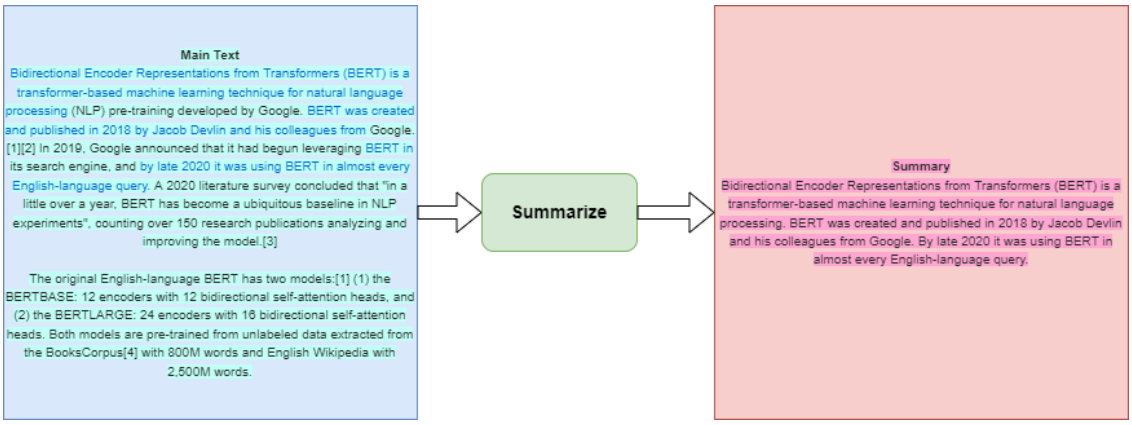


Figure 1 Summarization example

There are different types of summarizations the main two types based on output are:

1. **Extractive text summarization**

   This extractive text summarization method involves directly taking sentences from the text and incorporating them into the summary. This method produces a summary comprised of sentences and words directly retrieved from the text, meaning that the phrases in the summary are identical to sentences that appear in the text.

2. **Abstractive text summarization**

   Abstractive text summarization is a more advanced technique. This method requires not only that the algorithm understands the meaning of the sentence, but also the context of the sentence, and it should be able to modify the sentences and words as necessary. This technique seeks to generate a summary that is as concise and meaningful as feasible.

Other types of summarizations are based on input type single document, multi-document and based on the purpose generic, Query based, Domain specific.

What is sentiment analysis based on the news?

Using NLP (Natural Language Processing) tools, sentiment analysis determines if News is favourable means positive, neutral, or negative. News sentiment analysis assists companies and enterprises in monitoring article headlines and content for the sentiment. In News sentiment analysis, a news article and its content are analysed to determine a sentiment - favourable, negative, or neutral. This helps us to comprehend the public's perception of your company.

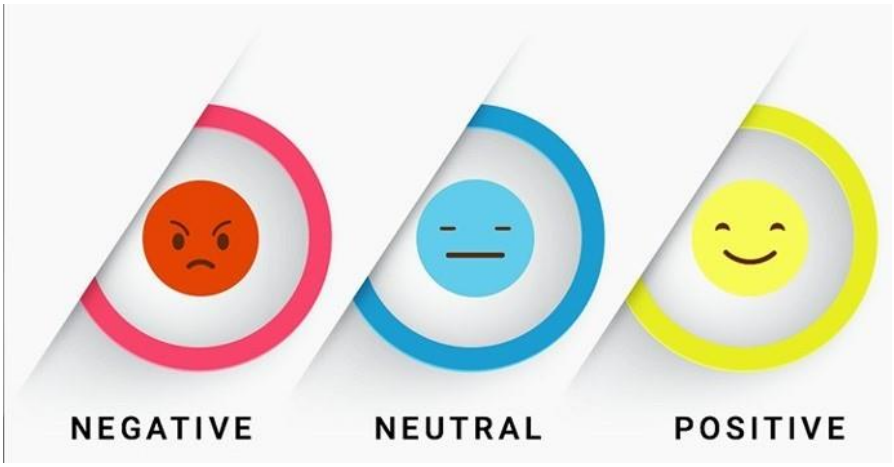


Figure 2 Sentiment emojis source (Symeon Symeonidis)

News sentiment analysis is crucial because it enables traders to make their decisions firmly. By categorising the mood behind news headlines traders can make their judgement more quickly and intelligently.

1.2: Motivation

Nowadays peoples got more aware about stock market and all other investing tools. A bunch of news related to stocks or sectors got released on daily basis which makes it difficult to read all of them in short time span because in live trading sessions based on single news only lot of price deflection may occur.

Moreover, due to the speed of our everyday lives, peoples frequently seek to maximize their time efficiency. Therefore, the purpose of our system is to keep readers abreast of the business news in an effective manner. Herein lies the possibility for machine learning to generate these summaries automatically and give sentiment as well. By building a model that can extract key sentences from an article, we may decrease the effort for peoples while also providing the news sentiment based on the news headline.

In this project, we focused more on the two tasks first one is abstractive summarization and the second one is sentiment analysis based on news and then we deliver a Web API that will show us the results.

1.3: Research Aim

This project aims to develop a system that gives us well-readable, concise, more relevant summaries from the news articles and gives their sentiment based on the headlines of the news. Our project focuses more on the following research questions:

- How can we combine two different tasks i.e., summarization and sentiment analysis as well as to deliver an API which shows us nice summary and sentiment of that news?
- Which methods can we use for our tasks?
- How the performance of the model should be evaluated?
- What can be the limitations of our system?

First, we aim to develop an extractive text summarization system capable of reducing texts of arbitrary length to a constant size that can be processed by the T5 model. Conditionalization of the pre-trained language model and explicit modelling of the local dependencies are required to enhance the model.

1.4: Methodology

This section shows the overall methodology of our project. In this project, Python was used to develop the system, which employs the Deep Learning framework PyTorch and the web application framework Flask. Here we used a multi-modal approach for both tasks. For this project, we used the BBC News summary dataset for summarization and two types of datasets for sentiment analysis. All these datasets are available on Kaggle and free to use. We used pre-trained deep learning models T5 and BERT and retrained them on our dataset. We fine-tuned them on our dataset to get the best possible results. We evaluated our model using different scores such as Rouge for summarization and F1 score for sentiment analysis. Then we created an API using the flask framework and integrated it with the newly trained model. Finally, we checked the results of our trained models the on-web app for unknown news and their headlines.

1.5: Project outcome

We successfully delivered the web application on a local host which gives us a good summary and shows us the sentiment of the news according to our models.

1.6: Outline of chapters

In Chapter 2, some literature review of both the tasks has been given with some techniques previously used for tasks.

Chapter 3 introduces some background information or knowledge about some techniques, models, and libraries used.

In Chapter 4, we have described the datasets which we have used for this project.

In Chapter 5, the actual implementation of models and web app creation are mentioned.

In Chapter 6, we have mentioned the evaluation techniques in deep and how did we use that for both the s.

In Chapter 7, the results generated from the models are discussed.

In Chapter 8, the discussion session we have given limitations and future work that can be done.

In Chapter 9, the conclusions are drawn according to our results.

CHAPTER II: Literature review

In 1958, the task of summarization utilising NLP initially appeared. At first for the task of summarization different statistical approaches were introduced such as TF-IDF [1] which computes scores of every sentence according to the frequency of words repeated and gives us the summary. This was more like a phrase extraction technique. Later, some machine learning approaches were introduced by different authors as given in [2] like the cluster-based method where a sequence of statements relating to the computed clusters can be used to construct the clusters, graph-based method or graph theoretic approach where there is a node for every statement and two sentences are related by an edge if they share some common terms, or if their similarity exceeds a certain threshold [2].

Some other approaches are, in [3] author utilizes a combination of Restricted Boltzmann Machine (RBM) and Fuzzy Logic to extract meaningful and lossless summary phrases from the text. In this study, fuzzy logic and RBM are employed as an unsupervised learning technique to improve the summary's precision. In this study, author discovered that the proposed method generates short summaries free of extraneous words. Utilizing elements such as Sentence-Centroid similarity and thematic phrases has improved the coherence between the sentences.

Extractive summarization is a summary that consists solely of extracted content so that the output of summary sentences are lines or words taken from the original text [4]. The majority of extractive summarizations fall into three kinds, construct an intermediate representation of the source text in order to express the text's key elements, based on the representation, grade the sentences, choose a summary based on the highest-scoring sentences [5]

A lot of previous work has been already done in summarization using NLP that includes Gensim which is open-source vector space toolkit based on text rank algorithm to generate summaries. Various python libraries are also there which summarizes our text like NLTK, and Sumy. But now a days transformer-based approach is widely used to generate summaries. Like BERT, T5 transformers gives nice result [8]. We can fine-tune these transformers according to our problem. BERTSUM is a version of BERT, fine-tuned for the task of text summarization. BERTSUM augments BERT’s architecture, by adding summarization layers that can utilize the context vectors for words (BERT’s output) and generate context vectors for sentences, at large [9]. To improve model accuracy, some abstractive summarization projects have taken principles from the reinforcement learning (RL) field. For example,[6] [7] suggested a hybrid extractive-abstractive architecture that hierarchically uses two neural networks to choose important lines from a source using an RL-guided extractor and then rewrites them abstractly to provide a summary.

In recent years, sentiment analysis has been intensively investigated utilising a range of machine learning techniques, such as natural language processing (NLP). Traditional NLP techniques have the drawback of not working well with extended phrases and specialised information. However, these issues can be resolved by adding recurrent deep learning models and fine-tuning big language models. Several authors have employed traditional machine learning methods, such as support vector machines [10] and tree-based models [11], to address concerns concerning the prediction of the stock sentiment of financial market news or social media content. One of the primary NLP approaches utilised in financial forecasting is sentiment analysis [12], which involves the interpretation and categorization of emotions within various text data sources.

Intuitively, the fluctuation of stocks can be attributed to the collective actions of stockholders in response to news [13]. In [14] the author utilised a polarity detection algorithm to initially categorise news articles and construct the train set. This method utilised a dictionary-based approach. The dictionaries for positive and negative terms are constructed using both general and finance-specific

words with emotional connotations. The author built their own dictionary for the removal of stop words, which includes finance-specific stop terms. They constructed three categorization models based on these data and evaluated them under various test conditions. Then, after comparing their results, Random Forest performed exceptionally well for all test situations, with an accuracy range from 80% to 90%.

Zhang et al. [15] conducted a literature assessment on sentiment analysis using deep learning algorithms. According to this study, word embeddings are the standard strategy for text representation in contemporary methodologies [16]. The models that learn most frequently are recurrent neural networks such as LSTM [17] and GRU [18]. In addition, the number of studies utilising the Attention Mechanism [19] in sentiment analysis increases. Studies that execute a welcome combination of embeddings, bidirectional methods, and attention mechanisms are beginning to surface in the scientific literature, outperforming numerous state-of-the-art algorithms. In [20] the author employs a BERT-based sentiment model to determine if news is positive, negative, or neutral. They performed this experiment on 582 financial news articles which are manually labelled. It outperforms all other approaches.

CHAPTER III: Background

3.1 Natural Language Processing

Natural language processing (NLP) is a branch of computer science and artificial intelligence that enables computers to comprehend and process human language. This involves the comprehension of both written and spoken language, which presents numerous challenging obstacles. Today, computer apps are expected to translate, respond to voice instructions, provide directions, and even generate texts that resemble human writing. Due to the abstract and contradictory nature of the human language, these obstacles are difficult for computers to overcome. Examples include humour, sarcasm, irony, intent, and feeling, which vary not only between languages but also between individuals.

NLP seeks to address these issues by transforming language into numerical computational inputs that a computer can comprehend and process. Combining computer algorithms with statistics enables the extraction, classification, and labelling of language.

3.2 What is transfer learning?

In machine learning, the application of a previously learned model i.e. pretrained model to a new problem is known as transfer learning. In transfer learning, a machine applies the knowledge acquired from a previous assignment to improve its prediction of a new task. When training a classifier to predict if an image contains food, you may, for instance, use the information learned during training to distinguish beverages.

It offers a few benefits, the most important is that it reduces the training time for our new model, improved network performance and does not require the large amount of data.

3.3 Transformers

Encoder and decoder are the two key components that make up a transformer's attention-based architecture. Vaswani et al. [20] presented the model to overcome existing issues with recurrent models that prevent parallelization, which would result in a longer training period and a performance loss for longer dependencies. Due to the non-sequential, attention-based nature of transformers, they can be massively parallelized while maintaining constant sequence and path complexity. Transformers are utilised to address translation issues.

Encoder: The encoder is composed of numerous encoder layers, with two sublayers per layer. The first sub-layer is a mechanism for multi-headed self-attention. Self-attention signifies, for instance, that the target sequence is identical to the input sequence. In other words, self-attention is merely another type of attention mechanism that links distinct input sequence positions. The phrase "multi-head" refers to the use of scaled dot-product attention, which enables the parallel processing of numerous attention computations [20]. The second sublayer is a basic position-wise feed-forward network with full connectivity. Each sublayer possesses both a residual connection and a layer normalization. The objective of this is to combine each sub-output layers with its preceding input. Figure 3, left block, depicts the encoder of a transformer.

Decoder: The decoder has a similar structure as the encoder. Masked multi-head attention is a modified multi-head attention mechanism that, unlike self-attention, stops it from attending to future positions. The purpose of masking places is to ensure that the predictions made do not forecast the target sequence's future [20]. Figure 3, right block, depicts the decoder of a transformer.

For instance, in translation tasks, the encoder is given words in a specific language while simultaneously processing each word. It then generates embeddings for each word, which are vectors that represent the numerical meaning of the words. Similar words have vector numbers that are closer together. Combining the embeddings from the encoder and the previously created translated sentence allows the decoder to predict the translation of a word in a sentence. The decoder guesses each next word till the end of the phrase.

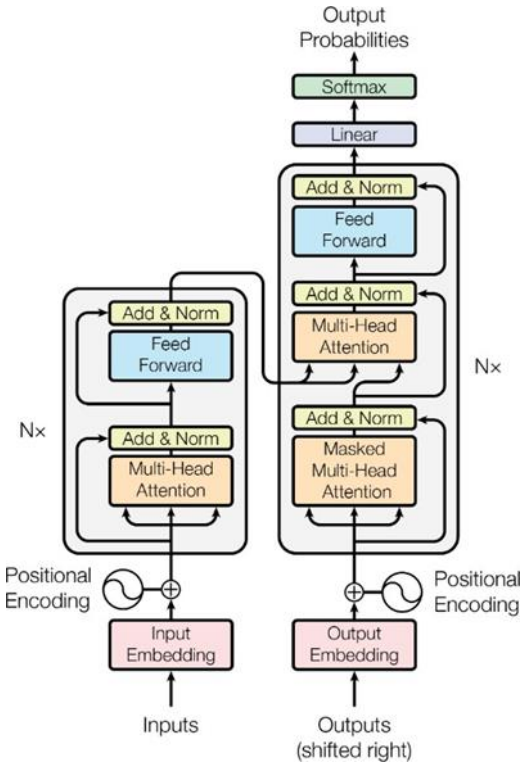


Figure 3 Model architecture of a transformer illustrated by Vaswani et al. [20]

Transformers are limited to 512 tokens. The memory and compute requirements of a transformer increase quadratically with sequence length, making the processing of large sequences impracticable [20]. This indicates that input for transformers must be less than 512 tokens. Later, additional techniques such as Transformer XL, which uses a recurrent mechanism to process text sequences larger than 512 tokens [21] were introduced. In the majority of instances, it is necessary to truncate sequences longer than 512 tokens in order to make them fit.

In general, the encoder understands the meaning of a word by studying its origin, grammar, and, most importantly, context. On contrary, the decoder learns the linguistic relationship between the original word and the target word.

3.2 Models used

3.2.1 BERT

BERT is an open-source framework for machine learning and natural language processing (NLP). BERT is aimed to assist computers in understanding the meaning of ambiguous words in the text by establishing context from the surrounding text. The BERT framework was pre-trained with Wikipedia text and can be refined using question and answer datasets.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based on their connection. (In NLP, this process is called attention.)

BERT is a transformer-based model developed by Devlin and colleagues [22]. The authors argue that prior language representation models, such as RNNs, were limited in encoding tokens since they only considered tokens in one direction. In contrast to RNNs, the authors create a Bidirectional Encoder Representation from Transformers (BERT) that can encode a token utilising tokens from both directions using transformers.

BERT is capable of resolving a variety of issues, including question answering, sentiment analysis, and text summarising. Nonetheless, a pre-training phase and a phase of fine-tuning are necessary to tackle these language-related issues. First, BERT is pre-trained to comprehend language, and then it is fine-tuned so that it can learn to perform a certain task.

Input and Output Embeddings

Similar to previous language models, BERT embeds each input token in a vector representation using a token embedding layer. In addition, BERT possesses two more embedding layers known as segment and position embeddings. The figure below displays an illustration of the three embedding layers.

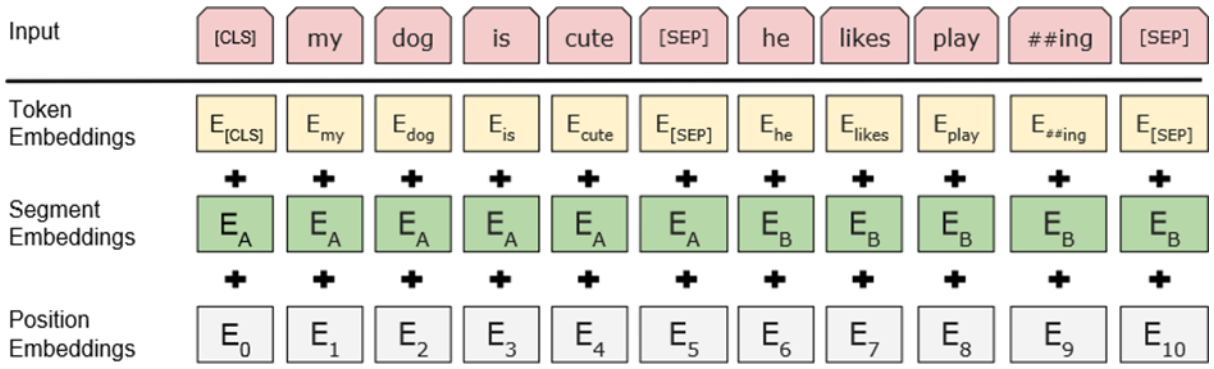


Figure 4 The input embeddings and embedding layers for BERT discussed by Devlin et al. [22]

To construct an input embedding, the Token, segment, and position embeddings for a specific token are added together. Before the embedding layers process the input, WordPiece is used to tokenize the input text.

WordPiece

BERT adopts Wu et al [23] .'s WordPiece tokenization proposal. The use of WordPiece tokenization was intended to improve the handling of uncommon words i.e., rare words. In this Words were broken

down into sub-words (WordPieces) using a predetermined vocabulary set. According to BERT, its vocabulary size is 30,000 Word-Pieces. When a word's rarity increases, it can be separated into individual characters. In addition, all subwords save the first subword of a word are represented by "##" characters. Because they may be repetitive for the entire term, the initial subword is split from the other subwords. For example, "Swimming" can be separated into "Swim" and "##ming." The subword "swim" imparts significance to swimming because the two words are closely linked.

BERT is pre-trained on two distinct but related NLP tasks employing this bidirectional capability: Masked Language Modelling and Next Sentence Prediction.

Masked Language Modelling

The goal of Masked Language Model (MLM) training is to hide a word in a sentence and have the algorithm predict (mask) the hidden word based on its context. Masked Language Modelling (MLM) is a task performed unsupervised during the BERT pre-train. MLM aims to aid BERT in comprehending deep bidirectional representations. MLM is performed by randomly masking 15% of all WordPiece tokens in the input sequence. The tokens are masked by substituting them with the [MASK] token, which BERT recognises and forecasts.

Next Sentence Prediction

The purpose of Next Sentence Prediction training is to teach the computer to predict whether two provided sentences have a logical, sequential relationship or whether their relationship is merely random. Next Sentence Prediction is an additional unsupervised task performed during the BERT pre-train. This job requires identifying the relationship between two statements. BERT is pre-trained for a binarized next sentence prediction that may be derived from any monolingual corpus [22] in order to capture sentence associations. This is accomplished by assigning fifty percent of the inputs to sentence pairs in which the second sentence is the next sentence from the corpus. The remaining 50% contain identical sentence pairs, but the second sentence is a random selection from the corpus. For example, if A is a statement from the corpus, then B is the subsequent statement of A 50% of the time and a random sentence from the corpus the other 50% of the time.

Pretrained BERT models

We can employ BERT for any downstream task by using language models that have been pretrained in two ways: feature-based and fine-tuning [22]. As mentioned in the preceding section, a BERT model must be pre-trained before to being fine-tuned on various tasks since it must be taught to encode language. This procedure is time- and resource-intensive. For instance, Devlin et al. [22] pre-trained the BERT model using four cloud TPUs for four days (16 TPU chips in total). Consequently, numerous BERT models are distributed as pre-trained models with initialised parameters that are prepared for specific tasks. The model that has been pre-trained can then be fine-tuned for specific purposes.

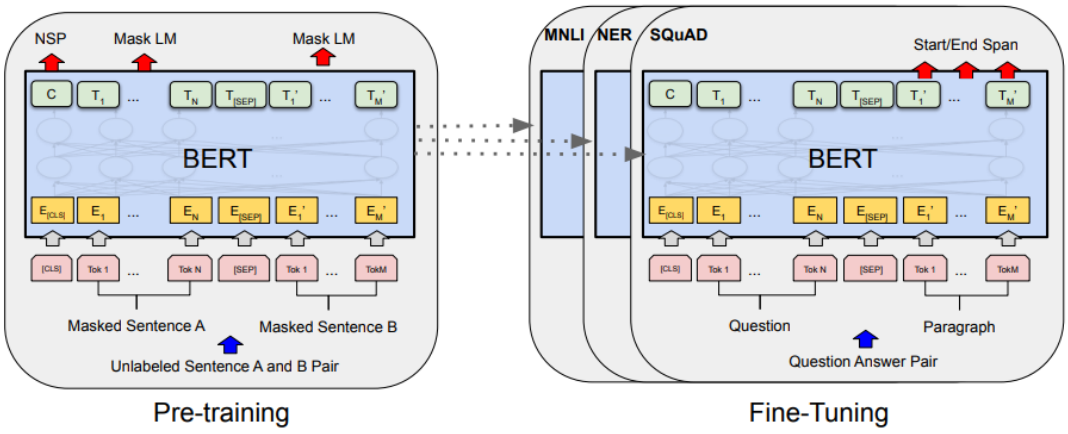


Figure 5 Fine tuning and pre training of BERT [22]

The figure above depicts the model's pretraining and fine-tuning process; for both tasks, the same architecture is employed, apart from the output layer. Due to this, we can use the same model that has been pretrained for various downstream tasks by simply fine-tuning it.

Some downstream tasks are:

Sequence-to-sequence-based language generation tasks such as:

- Question answering
- Abstract summarization
- Sentence prediction
- Conversational response generation

Natural language understanding tasks such as:

- Word sense disambiguation
- Natural language inference
- Sentiment classification

3.2.1 T5 model

T5 [24] is a "more recent" transformer-based PLM. T5 can be considered a solid foundation for NLP activities. Text-to-Text Transfer Transformer (T5) is a transformer model with an architecture quite similar to BERT [22]. The model is pre-trained on the Colossal Clean Crawled Corpus and can do several NLP tasks such as translation, question answering, summarisation and classification. The model employed here takes the properties of a pre-trained model and applies its knowledge to the data used here without fine adjusting the model.

The T5 Small model consists of 12 blocks, in both the encoder and decoder. Each block's feed-forward networks consist of a dense layer with 2048 output dimensions, followed by a ReLU nonlinearity and another dense layer. Each and every attention mechanism has 8 heads. All additional sublayers and embeddings are 512-dimensional [19]. The T5 model is available in several sizes, the smallest of which has almost 220 million parameters. This work used the T5 Small model, which includes sixty million parameters. In addition, models with 770 million (T5 Large), 3 billion (T5 3B), and 11 billion (T5 11B) parameters are available. The transformers Python library makes use of the pre-trained models.

Colossal Clean Crawled Corpus (C4), which was constructed for T5's pre-training, was also presented in [19]. The Common Crawl is a repository of Internet-extracted text. This corpus containing just English words is the outcome of sifting and cleaning this archive. This dataset is accurate and substantial in size (about 750GB). T5 was created as a sequence-to-sequence model, and its architecture is comparable to that of [20]. Similar to BERT, the designers of T5 opt to adopt a pre-training denoising target. In the experiment, different pre-training objectives (Denoising, Language Modeling, and Deshuffling) were exhaustively evaluated using a variety of parameters (including different span length, corrupting rate, etc.). In addition, the target with the highest performance was to replace token spans with indexed placeholders and make the decoder forecast the substituted tokens.

In contrast to BERT, the pre-training and fine-tuning processes were merged and presented in a sequence-to-sequence format in T5. The encoder received input for all tasks (translation, emotion analysis, sentence matching, etc.) and the decoder generated the targets. Therefore, the goals of some datasets were modified to conform to the seq2seq output style. Translation and summarization, which were already in sequence-to-sequence format, required no modifications to the targets. Also, similar to GPT-2, task-specific prefixes were added to the input sequences during fine-tuning for various tasks, providing the model with task-specific information.

Despite the fact that T5's architecture closely resembles the original transformer sequence-to-sequence approach [20], T5 utilised a very similar architecture. There were various alterations made to it:

- T5 utilised a streamlined form of layer normalization that lacked an additive bias.
- The layer was normalized before to the residual connection.
- A relative position embedding rather than a fixed position embedding was employed to represent the positions of tokens.
- The token embedding weight was shared between the encoder and decoder.

T5 is a sequence-to-sequence model, and it has an auto-regressive decoder. So, in nature, it coheres with the task of language generation. For this project, a T5 model can be fine-tuned with the article and summary data formulated into inputs and targets. For sentence decoding, T5 uses different techniques such as grid search algorithm, beam search algorithm, top k sampling, etc. For our task of summarization, we used beam search because it gives improved performance for longer output sequences [19].

Beam Search algorithm

Based on conditional probability, the beam search algorithm finds numerous tokens for a point in each sequence. The algorithm can take any number of N best choices using the Beam width hyperparameter. In greedy search, we just choose the best word for each location in the sequence, however here we expand our search or "width" to include alternative words that may be a better fit. With Beam search, we additionally take the N best output sequences and compare the probability of the current previous words to the current place in the sequence we are decoding.

3.3 Optimizer

ADAM: Adam [42] is a stochastic objective function gradient-based algorithm with adaptive estimations of lower-order moments. Adam is one of the most recent cutting-edge optimization techniques employed by many machine learning practitioners. The direction of the update is determined by the first instant normalised relative to the second moment.

Adam's update rule:

$$\theta_{n+1} = \theta_n - \frac{\alpha}{\sqrt{\hat{v}_n + \epsilon}} \hat{m}_n$$

3.4 Evaluation Metrics

Evaluation is essential to the development of the summarising process as well for analysing sentiments. Here for the summarization task, we used the Rouge score and for the sentiment analysis task, we used the F1 score for evaluation.

ROUGE:

Rouge stands for **Recall Oriented Understudy for Gisting Evaluation**. It is a set of metrics given by Lin [25] in 2004 for the automatic evaluation of summaries generated by different models. The Rouge metrics compare model-generated summaries to one or more human-generated reference summaries. The ROUGE-1, ROUGE-2, and ROUGE-L metrics are extensively used for benchmarking document summarising algorithms, such as on the CNN/Daily Mail dataset's leader board for document summarization [26]. Typically, the recall, precision, and F1 score are calculated for each metric. With ROUGE, the genuine positives are the terms within the reference summary' phrases.

ROUGE-N:

ROUGE-N is defined as the overlap of candidate summary and reference summary n-grams. As previously stated, the most used metrics are ROUGE-1 and ROUGE-2, where ROUGE-1 refers to the overlap of unigrams (single words) and ROUGE-2 refers to the overlap of bigrams (two or more words) (two adjacent words).

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (1)$$

Figure 6 Rouge N formula [26]

ROUGE-L: Longest Common Subsequence

Longest Common Subsequence (LCS) of terms shared by a candidate summary and a reference summary. It represents sentence-level similarity based on the longest word-sequence matches.

ROUGE-W: Weighted Longest Common Subsequence

The longest common subsequence with ROUGE-L has the disadvantage of not favouring consecutive matches. This disadvantage indicates that word sequences with a smaller geographical difference will have the same ROUGE score as sequences with a bigger spatial difference. ROUGE-W addresses this issue by identifying the length of consecutive matches encountered and providing a weighted longest common subsequence.

ROUGE Limitations

Despite ROUGE's prominence among studies on automatic text summarization, its shortcomings must be addressed:

- ROUGE evaluates just the selection of content and no other aspects such as fluency and coherence.

- ROUGE relies solely on precise overlaps, although a summary can provide the same information as an article using other terms and synonyms.

- ROUGE was originally intended for use with numerous reference summaries, taking into account the subjectivity of summaries.

There is another evaluation metric called BLEU which can also be used to evaluate summaries. In BLEU, we value precision: how many n-grams from the candidate translation appear in the reference? In ROGUE, recall is important: how many n-grams from the reference summaries are present in the candidate? As we are implementing extractive summarizer ROUGE is the proper metric which gives us total Rouge score based on n-grams present on both reference summary and model summary.

3.5 Development Environment

A development environment or programming environment is a combo of a text editor and code implementation. Here we used google colab, Flask, HTML, and Bootstrap. The majority of this project is based on Deep learning technology. This project will utilise Google Colaboratory (CoLab), a free Jupyter notebook interactive development environment (REPL) hosted in Google's cloud.

Google colab: Google Colab is a document that enables you to write, execute, and share Python code in a web browser. It is a variant of the popular Jupyter Notebook that is part of the Google toolkit. Jupyter Notebooks (and consequently Google Colab) allow you to produce a document including executable code in addition to text, photos, HTML, LaTeX, etc., which is then kept in your Google Drive and can be shared with peers and co-workers for editing, commenting, and viewing [27].

Flask: Flask is a web framework and Python module that facilitates the development of web applications. It has a minimal and extensible core: it's a microframework that lacks an Object Relational Manager (ORM) and similar functionalities. It has several cool features, such as URL routing and a template engine. It is a framework for WSGI web applications. The Web Server Gateway Interface (WSGI) has been adopted as a standard for the creation of Python web applications. WSGI specifies a standard interface across web servers and web applications.

HTML: Hyper Text Markup Language is the programming language that enables the building of websites. The language, which contains code words and syntax much like any other language, is very simple to comprehend and, as time passes, enables ever-more-powerful creations. Under the guidance of the World Wide Web Consortium, the group that defines and maintains HTML, the language continues to evolve to meet the demands and expectations of the Internet; for instance, with the move to Web 2.0.

3.6 Python Libraries imported

Python is an interpreted, object-oriented programming language like PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a few operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 9,8. Python was created by

Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus.

Python can be used as the script in Microsoft's Active Server Page (ASP) technology. Python is used to create the scoreboard system at the Melbourne (Australia) Cricket Ground. Z Object Publishing Environment, a popular Web application server, is also written in Python languages.

PYTHON PLATFORM

Apart from Windows, Linux, and macOS, Python implementation runs on 21 different platforms. Iron Python is a .NET framework-based Python implementation, and it is capable of running in both Windows, Linux and in other environments where the .NET framework is available.

PYTHON LIABRARY

In the past, Machine Learning jobs were performed by manually coding all algorithms and mathematical and statistical formulas. As a result of numerous Python libraries, frameworks, and modules, it is now significantly simpler and more efficient than in the past. Python is currently one of the most popular programming languages for this task, and it has supplanted many other languages in the business, in part due to its extensive library collection. These are the Python libraries utilised in our project:

- NumPy
- Scikit-learn
- PyTorch
- Pandas
- Matplotlib
- Transformers
- Rouge

NumPy:

NumPy [32] is a widely used Python toolkit for processing massive multidimensional arrays and matrices using a vast range of high-level mathematical operations. Machine Learning uses extensively for fundamental scientific computations. It is very helpful for linear algebra, Fourier transform, and random number functions. Internally, advanced libraries such as TensorFlow employ NumPy to manipulate Tensors.

Scikit:

Scikit-learn [38] is one of the most widely used machine learning packages for traditional ML methods. It is based on two fundamental Python libraries, namely NumPy and SciPy. The majority of supervised and unsupervised learning algorithms are supported by Scikit-learn. Scikit-learn may also be used for data mining and data analysis, making it an excellent tool for ML beginners. This is used to divide our dataset into train and test sets.

PyTorch:

PyTorch [33] is a prominent open-source Machine Learning library for Python that is based on Torch, an open-source Machine Learning toolkit written in C with a Lua wrapper. It offers a vast array of tools and libraries that assist Computer Vision, Natural Language Processing (NLP), and numerous other ML

applications. It enables developers to do GPU-accelerated computations on Tensors and aids in the creation of computational graphs.

Pandas:

Pandas [39] is a popular Python data analysis toolkit. It has no direct relationship with Machine Learning. As we know, dataset preparation is required before training. Pandas is useful in this situation because it was designed primarily for data extraction and preparation. It features advanced data structures and a multitude of data analysis capabilities. It has numerous inbuilt ways for data grabbing, merging, and filtering.

Matplotlib:

Matplotlib [40] is a popular Python data visualization package. Similar to Pandas, it is unrelated to Machine Learning. It is especially useful when a programmer wants to visualize patterns in the data. It's a library for making 2D plots and graphs. A tool called pyplot simplifies charting for programmers by providing options to customise line styles, font attributes, axis formatting, etc. It offers numerous graphs and plots for data visualisation, including histograms, error charts, bar charts, etc.

Transformers:

Transformers [41] is a library that contains different pre-trained models such as BERT, and T5 which we can fine-tune later for different tasks. It provides APIs for downloading and training cutting-edge pre-trained models with ease. Using pre-trained models can minimize your computing expenses, carbon footprint, and training time.

Rouge:

Rouge is a python library that is used to check the quality of our generated summaries from our trained model. It gives us rouge scores as R1, R2, and RL scores [25].

CHAPTER IV: Dataset Description

For this project we have used three different datasets one is for the news summarization task the and other two the for sentiment analysis task they are as follows:

1.1: BBC News Summary dataset (Kaggle)

- This dataset contains near about 4450 news articles in different domains such as business, entertainment, politics, sports, and tech along with their human-generated summaries [28].

- For this project particularly we have used business news only which contains 510 news articles with their summaries.

- Five hundred ten business news stories from the BBC from 2004 to 2005 are included in this dataset for extractive text summarization in the News Articles folder. The Summaries folder has five summaries for each article. The title of each article appears in the first clause [28].

	Unnamed: 0	File	News	Summary
0	0	361.txt	US consumer confidence up\n\nConsumers' confid...	Wal-Mart, the largest US retailer, has said it...
1	1	245.txt	The 'ticking budget' facing the US\n\nThe budg...	Brute force budget cuts or spending caps would...
2	2	431.txt	Mitsubishi in Peugeot link talks\n\nTrouble-hi...	Trouble-hit Mitsubishi Motors is in talks with...
3	3	141.txt	BMW reveals new models pipeline\n\nBMW is prep...	Typically it takes about three years from when...
4	4	487.txt	World leaders gather to face uncertainty\n\nMo...	More than 2,000 business and political leaders...

Figure 7 BBC News summary dataset

- This is the Business news summary dataset which contains four different columns. We pre-processed this dataset for our task of summarization.

Columns	Description
News	This column contains the actual whole articles of the news.
Summary	This column contains the human-generated summaries of the news articles

Table 1 BBC news summary dataset description in short

1.2: Financial sentiments for Multi-entity News Headlines

- This dataset is publicly available as SEntFiN 1.0, a human-annotated dataset containing 10,753 news headlines with entity-sentiment annotations, of which 2,847 headlines contain multiple

entities, frequently with contradictory feelings [30]. The author enhances this dataset with a database of over one thousand financial firms and their numerous news media representations, totalling over five thousand terms [29].

- This dataset is labelled as Positive news, Negative news, and Neutral news.

	S No.	Title	Decisions	Words
0	1	SpiceJet to issue 6.4 crore warrants to promoters	{"SpiceJet": "neutral"}	8
1	2	MMTC Q2 net loss at Rs 10.4 crore	{"MMTC": "neutral"}	8
2	3	Mid-cap funds can deliver more, stay put: Experts	{"Mid-cap funds": "positive"}	8
3	4	Mid caps now turn into market darlings	{"Mid caps": "positive"}	7
4	5	Market seeing patience, if not conviction: Pra...	{"Market": "neutral"}	8

Figure 8 Financial sentiments for Multi-entity News Headlines dataset

- We use this dataset to for sentiment analysis based on news headlines for our system as this dataset is specially labelled on news related to the stock market.
- It contains four different columns out of which we used only Title and Decisions column for our task.

1.3: Sentiment Analysis for Financial News

- This dataset (FinancialPhraseBank) contains the sentiments associated with financial news items from the perspective of an individual investor [31].
-

	sentiment	statement
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...

Figure 9 Sentiment Analysis for Financial News dataset

- It contains about 4500 financial news along with their labels as Positive, Negative and Neutral.
- Here we used two datasets for sentiment analysis so that we can check model performance on two different labelled datasets.

CHAPTER V: Implementation

In this section, we will see the implementation of T5 base extractive news summarization model and sentiment analysis model using pretrained BERT. Firstly, in section 5.1 we will see some data pre-processing techniques for both the tasks. Then, in section 5.2, model implementation is described. Lastly, in sections 5.3, and 5.4, fine-tuning of the T5 base model and BERT model, and prediction are scribed.

5.1 Data pre-processing techniques used and EDA

Computer science and artificial intelligence's field of study known as "natural language processing" (NLP) focuses on giving computers the ability to comprehend and analyse human language. This involves comprehending spoken and written language, which presents a variety of intricate difficulties. To perform any NLP project some basic data pre-processing techniques are used as follows:

5.1.1 Tokenization

Tokenization is nothing but the process where we separate the whole sentences into different tokens which helps the machine to better understands the text data. It is the most basic step first to be done in NLP. For example, “Yes you did it, Nice!” it is one sentence when we apply tokenization it will tokenize the sentence into separate words like “Yes”, “you”, “did”, “it”, “,”, “Nice”, “!”.

5.1.2 Stop-word removal

Stop words are the words that do not contribute any meaning to the texts or news, so we consider removing them. There are different stop words in English language such as “the”, “a”, “an”, etc. These words do not give any significance, so we discarded them.

5.1.3 Stemming

Stemming is a technique where we separate and remove the stem word from its suffix word. For example, “magically” is a whole word for which “magic” is the stem word. It’s a text normalization technique.

5.1.4 Word Embedding

Word embeddings are a method for expressing words that captures both syntactic and semantic information. Typically, words are mapped to a fixed-dimensional vector space of numbers, where words with similar meanings are clustered together. This allows for the detection of synonyms and the suggestion of additional words for sentences. Language models that employ neural networks to train and learn word connections from a huge corpus of text obtain and use word embeddings.

5.1.5 Word cloud formation

We formed word cloud especially for sentiment analysis to check what are some words which give the strong probability of sentiment as follows:

Figure 10 Word cloud for the whole dataset

- The above figure shows the word cloud for our whole dataset.

Figure 11 World cloud for positive news

- The above figure gives us a word cloud for positive labelled news we can see some words like profit, gain, growth, bullish, etc. which gives us some idea about positive news.

Figure 12 Word cloud for Negative news

- The word cloud above shows us the most repeated words for Negative news some of these words are fall, drop, weak, slip, loss, etc. It gives us some intentions regarding news which may have negative impact.

Figure 13 Word cloud for neutral news

- The word cloud above shows us cloud for neutral news which may do not have any impact on stocks or sectors. It includes some common stock market terms which show us the neutral impact of that news.

5.1.6 Word count

We use a histogram-based graphical method here to check the average word count. It gives us an idea about how many words on average are present in our news articles as well as their summaries. The following figure shows the word count graph for our BBC business news summary dataset.

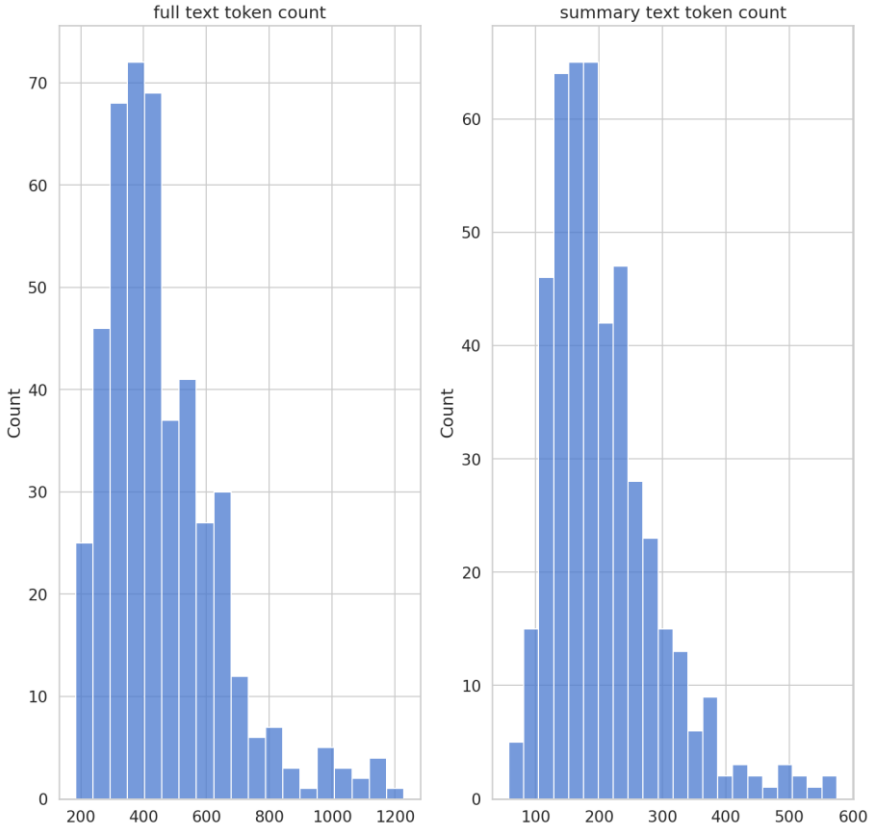


Figure 14 Token count graph of articles and summaries

From the figure above we can observe that on average total article length was 400 to 600 words per news article and for the summaries total summary length was between 100 to 200 words per news article summary.

5.2 Models

The 'T5 base' and 'Bert-base-uncased' models have been implemented for the two distinct tasks. By retraining these models on our labelled datasets and fine-tuning them, we were able to predict news summaries as well as sentiment based on their headlines.

Pre-trained models: Hugging Face introduced the pre-trained models utilizing the Transformers Python package. Transformers provides general purpose architectures for several workloads in both PyTorch and TensorFlow, with over 32 pre-trained models spanning a wide range of programming languages. Transformers' Auto Class feature provides a dynamic tokenizer and model initialization. Both T5 and BERT models were in the Hugging Face API and could be integrated using Transformers.

T5 base model:

We used the T5 base model here as described in section 3 for the task of summarization. For the tokenization task we the imported T5 tokenizer which directly tokenizes our news in the form of text. For text generation, we added T5forconditionalgeneration which does add one language model head to the base model. The T5forconditionalgeneration model lets us to generate text on basis of training of our T5 model. For this project, we have used the standard version of the model i.e., the T5 base model. It has 12 encoder decoder layers and about 220M parameters. Here we used the Adam optimizer described in section 3.

Why do we use the T5 Base model only for the summarization task?

The choice of the T5 base model is justified by the fact that it provides better generalization as compared to BERT. As our dataset contains 510 data points means news articles with their summaries it is a bit difficult to train this on BERT because BERT requires a larger dataset to train properly. Pre-trained T5 model gives nice results on the BBC news dataset it outperforms other models [7].

BERT model:

For sentiment analysis, we imported the ‘Bert-base-uncased’ model which is a classification model trained on Wikipedia texts. It classifies the news according to the labels we have provided i.e., Positive, Negative or Neutral. Here for tokenization, we used BertTokenizer to tokenize the all-news headlines.

Data loader:

Data is available in numerous formats and shapes. For loading and processing the BBC news dataset, we used data loader. Data loader implemented in such a way that it shuffles and splits the text data into train, validation set to match input format. We use data loader so that we can feed our data in right way to our deep learning model.

5.3 Hyper-Parameters

First, we need to set some hyperparameters for both the models before fine-tuning them

- **Epochs**: As we have 220M parameters fine tuning requires only a few epochs. Here for summarization, we choose **four** epochs for our experiment. Devlin et al. [22] conducted their fine-tuning studies using three epochs. For sentiment analysis as we are using BERT, we took only **one** epoch to train the model on our dataset.

- **Batch Size:** To maintain modest memory usage, Peltarion [34] advises keeping the product batch size X max sequence length at < 3000. Consequently, if the sequence length is set to 512, the batch size is approximately **eight** for the summarization task.

- **Learning rate:** In order to prevent catastrophic forgetting, which can occur when a model is fine-tuned, a modest learning rate was used. This means that the newly tuned model can forget the information it previously learned. By establishing a very modest learning rate on the order of 10^{-4}, this problem is eliminated. Here we choose the learning rate as **0.0001**.

5.4 Fine tuning of models

There are different ways to fine-tune the models such as

1. Feature extraction: We can use a model that has been pre-trained to extract features. We can delete the output layer (the one that provides the odds of belonging to each of the 1000 classes) and then use the entire network as a fixed feature extractor for the new data set [35].
2. Use of the architecture of the pre-trained model: What we can do is apply the model's architecture while initializing all weights arbitrarily and retraining the model with our dataset. [35]
3. Freeze all layers and train some: Partially training a pre-trained model is another method to utilize it. What we can do is freeze the weights of the model's first layers while retraining only the upper levels. We can do experiments to determine how many layers must be frozen and how many must be trained [35].

Here we used the second method to fine-tune the T5 base model, we have used model parameters (i.e. model configuration) to fine-tune the model for the summarization task. Some parameters are as follows:

- **Repetition penalty:** It's the parameter for repetition penalty. It values between 1 and infinity. If it's 1 that means no penalty. It penalizes the sampling works by discounting the scores of previously generated tokens. Here we took value 3 as repetition penalty.
- **Early stopping:** This parameter is set to True so that beam search is stopped when at least the number of beams sentence finished per batch.
- **Length specific:** We can set the specific length of the output sequence generated. Here we have selected max_length of 150 tokens as we observed average token lengths summaries are between 100 to 200.
- **Number of Beams:** For the decoding method we used the beam search algorithm here.
  It is the number of beams for the beam search algorithm. It values between 1 and infinity. The default value for this parameter is one. Here we have chosen number of beams as 3.

- **Length penalty**: Length penalty is exponential. 1.0 indicates no penalty. Set to values less than 1.0 to encourage the model to generate shorter sequences and to values greater than 1.0 to encourage the model to generate longer sequences.

As sentiment analysis is our secondary task, we have used a pre-trained BERT classifier directly to get sentiment based on news headline data.

Sr. no	Models (Tasks)	Time taken to fine-tune the models
1	T5 Base (Summarization)	7 min 45sec.
2	BERT (Sentiment analysis for dataset 1)	1 hr 25 min
3	BERT (Sentiment analysis for dataset 2)	2 hr 15 min

Table 2 Time taken to fine-tune models

Why do we choose these values for the model parameters?

We took these values on experimental basis. We did some experimentation by changing these values taking some range and by checking model outputs for every change. For example, during experimentation we increased the number of beams to 10, maximum length as 200, early stopping False, repetition penalty as 1 and checked the results. We observed that Rouge score we got was near about 30 to 40 for different summaries. For the values we used above we got nice rouge score.

5.5 Prediction

Finally, we have made predictions on our BBC News test dataset, obtained the summaries for each news articles present in test data and checked their Rouge score. For sentiment analysis, we have predicted sentiments for various news headlines from different online sources.

5.5 Design of Web Application

For web application creation we have used Flask app through which we have integrated our model with the web app we created. Here we have formed three different HTML web pages 1st one is to take the unknown news as input for the summarization task, 2nd one gives us summarized results plus it takes news headlines as input for sentiment analysis, and 3rd one gives us the result of the sentiment from the news headlines. We used GET and POST methods in a flask to get the input from the web process it and post it on the web app.

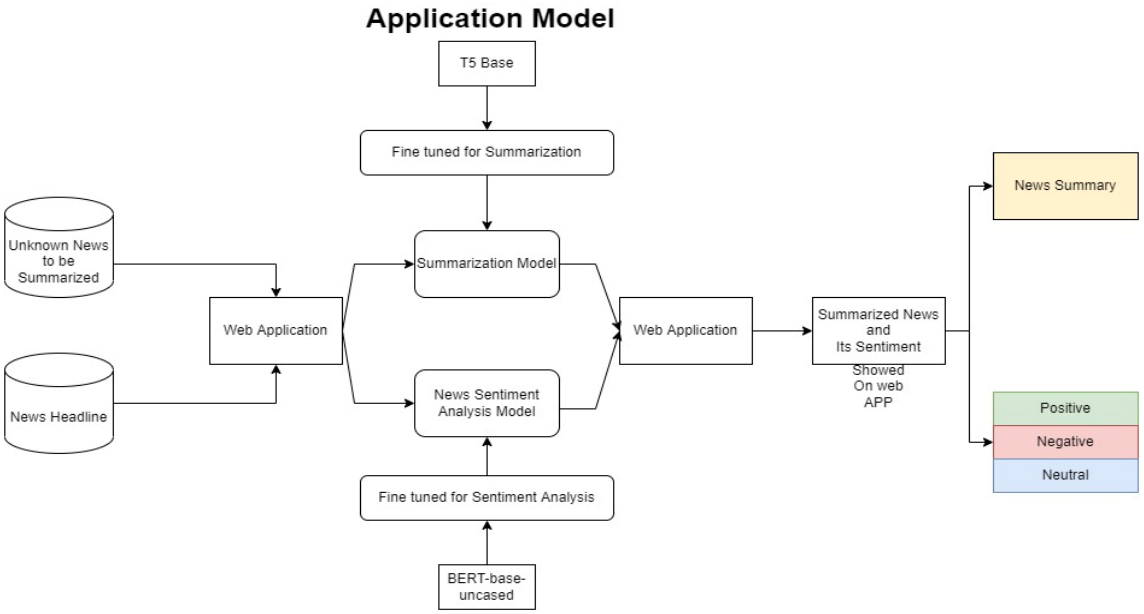


Figure 15 Our model architecture

In this project, the extractive summarisation and sentiment analysis system was constructed. The system was developed in Python using the Deep Learning framework PyTorch, and the web application framework Flask. The system was installed on a local-host Server so that it could be accessed by GUI. Figure 15 demonstrates the overall system design. Input to our web application is unknown news which we want to summarize and want to check its sentiment. Through flask app we have accessed our trained and fine-tuned models which get inputs from web application and gives us the results. The results given by the model is then sent back to the web application on which it got displayed through our HTML file. Similar procedure is done for both the task.

CHAPTER VI: Evaluation and testing

In this chapter, we will see automatic evaluation techniques or metrics Rouge used for our fine-tuned model. We will see how Rouge works for the automatic evaluation of summaries.

6.1 For summarization

Evaluation is an essential component of text summarization as well as sentiment analysis. In general, summaries can be assessed using either internal or extrinsic criteria. While intrinsic approaches aim to quantify summary quality through human review, extrinsic methods were used to determine the same using a task-based performance measure, such as the information retrieval-focused test. Evaluation techniques are helpful for assessing the usefulness and credibility of the summary. It is quite difficult to evaluate attributes such as comprehensibility, coherence, and readability. To determine the quality of a summary, a manually operated expert system is utilised. For the qualitative evaluation, the number of sentences selected by the algorithm that match the human gold standard is counted [36]. The ROUGE evaluator tool, which consists of a precision, recall, and F-measure, is used to quantify the summary's quantitative evaluation.

The Rouge package or library, which is a wrapper for the ROUGE summary assessment package, is the most often used software for computing ROUGE scores. At the time of this writing, Rouge only supports the English language. As we are also giving English news only to our model it is beneficial for us to use this.

Let’s see one example of how it calculates a score

ROUGE-N estimates the number of 'n-grams' that match our model-generated text and a 'reference' that is generated by humans. N gram is nothing but a group of tokens. Like 1 gram means it contains single word, bigram means it contains 2 consecutive words.

The N in ROUGE-N signifies the n-gram that we are utilising. For ROUGE-1, we would calculate the unigram match rate between our model results and the reference. ROUGE-2 and ROUGE-3 would respectively employ bigrams and trigrams. After determining which N to utilise, we determine whether to calculate the ROUGE recall, the precision, or the F1 score.

Recall: The recall measures the number of overlapping n-grams in the model output and the reference, then divides this number by the total number of overlapping n-grams in the reference [37]. It looks like this:

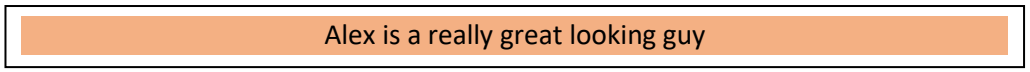
$$\frac{\text{number of n-grams found in model and reference}}{\text{number of n-grams in reference}}$$

$$\frac{\text{count}_{\text{match}}(\text{gram}_n)}{\text{count}(\text{gram}_n)}$$

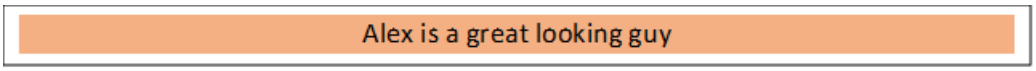
Figure 16 Recall formula (Source: TDS)

Let’s consider this example,

The machine produced summary be:



Reference summary be:



Here the number of overlapping words are 6 as well as total words in reference summary are 6

Therefore R = 6/6= 1.0

Precision: The precision metric is calculated very identically to the recall metric, except that instead of dividing by the reference n-gram count, we divide by the model n-gram count [].

$$\frac{\text{number of n-grams found in model and reference}}{\text{number of n-grams in } \textbf{model}}$$

Here Precision will be P = 6/7 = 0.86

F1-Score:

Now that we have both recall and precision numbers, we can compute our ROUGE F1 score as follows:

$$2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

For the summarization task, we split our dataset into 90% train data and 10% test data. We trained and fine-tuned our model on 90% of train data. We evaluate our model results on 10% data. As we have only 510 data points, we got 50 points to test our model result to evaluate our model summary.

Here we manually checked the results generated by our model one by one and calculated a rough score. The table below shows some Rouge scores as precision, recall, and F measures for random articles summaries i.e., references summaries and model-generated summaries.

	Rouge 1	*Rouge 2*	*Rouge L*
P	*0.59*	*0.53*	*0.60*
R	*0.88*	*0.82*	*0.89*
F	*0.724*	*0.64*	*0.72*
P	*0.44*	*0.36*	*0.45*
R	*0.50*	*0.44*	*0.51*
F	*0.47*	*0.39*	*0.474*
P	*0.58*	*0.35*	*0.54*
R	*0.65*	*0.42*	*0.58*
F	*0.62*	*0.38*	*0.56*
P	*0.44*	*0.34*	*0.45*
R	*0.66*	*0.61*	*0.67*
F	*0.53*	*0.43*	*0.52*

Table 3 Rouge score of some random News articles summaries from test dataset and model generated summaries of their main articles

Along with this Rouge score metric, we did manually evaluated summaries generated by our model for unknown news as well as for known news from our test dataset.

6.2 For sentiment analysis

Similar to the above task we split our sentiment analysis dataset into 90% training and 10% testing sets. Primarily, Training and Testing Accuracy is used to evaluate the pre-trained BERT model utilised in the research. In addition, Precision, Recall, and the F1 score are used to compare the performance of the model when applied to both balanced and imbalanced data.

Accuracy: Accuracy is the degree of a model's correctness. Typically, it is characterised by the ratio of right categories to the total number of classifications. Training precision is frequently used to assess the performance of a model during a single training epoch. The test accuracy is the model's accuracy once it has been completely trained.

Accuracy = (TP + TN) / (TP + FP + TN + FN)

TP represents True positives, TN represents True negatives, FP represents False positives, and FN represents False negatives.

Precision: Precision is characterized by the number of positive predictions successfully recognised by the model. Precision represents the accuracy of the model in estimating the Positive labels. When precision equals 1, the performance of the model is deemed to be optimal. The greater the precision, the fewer false positives.

$$\textbf{Precision} = TP / (TP + FP)$$

For sentiment analysis along with the two measures mentioned above, we have used the F1 measure to evaluate our model performance. F1 takes the weighted average of Precision and Recall into consideration when calculating the performance metric.

To test the model performance, we fed some random labelled news from the first dataset to the model trained on 2nd dataset, and vice versa, and manually checked the results generated by our models. In the next chapter, we will see some results.

CHAPTER VII: Results

7.1: News summarization result

90% of the experimental data from the dataset was utilised for model training. Furthermore, 10% of the data was used as the validation and test set. The model was trained for four epochs, and training is terminated when there are indications of overfitting, i.e. when the loss ceases to decrease.

As mentioned in T5's paper, a task-specific prefix was developed and appended to the input sequence. This provided the model with additional details about the task itself. Based on the model's performance, all hyper-parameters were changed. This is done to reduce bias from excessively adjusting the hyper-parameters and to improve the result on the test set. We used a tensorboard logger to display our loss. The figure below shows the validation loss got decreased with the time and number of epochs the best loss we got was **0.2936**.

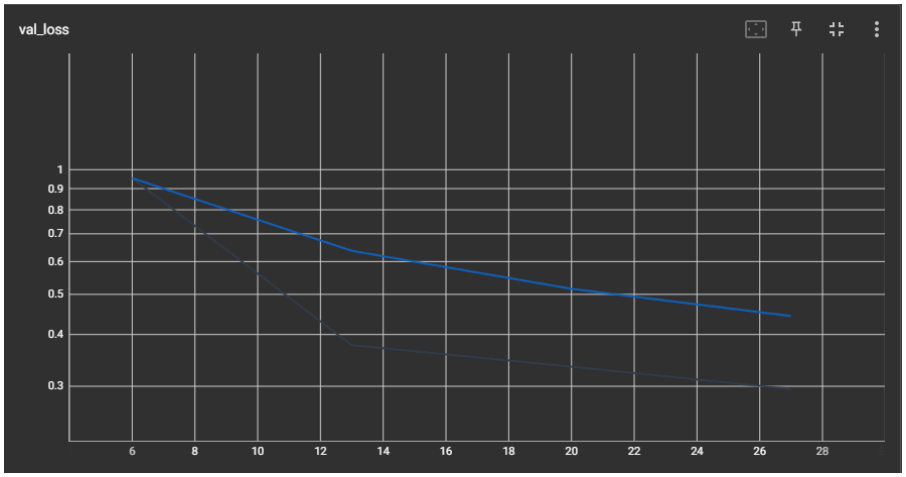


Figure 17 Validation loss

We have seen some Rouge scores we got from our model-generated summaries as compared to the reference summaries in the above section. If calculates the averages of all rouge scores we got, we can see that our model performs nicely on this dataset and produced some sensible, concise summaries. The table shows the average rouge scores we got from our model. Our model gives better results than state-of-the-art models.

Model	Rouge 1	Rouge 2	Rouge L
T5(On business news only) Average Rouge Scores	0.58	0.48	0.56

Table 4 Calculation of average rouge scores from our model generated summaries

In paper [7], author uses the same dataset for different models such as BART, PEGASUS and T5. The results from T5 are shown below

Model	Rouge 1	Rouge 2	Rouge L
T5(On whole news dataset)	0.47	0.33	0.42

Table 5 Rouge scores results from [7]

We can see that though we are using only business news our model outperforms the state-of-the-art results and provides us nice summary. Due to fine-tuning and choosing correct hyperparameters we got these results.

7.2: News sentiment analysis result

This section tabulates the results, including the confusion matrix and classification report, for the pre-trained model BERT utilised in this project.

Here first, we evaluated the efficacy of the fine-tuned model in predicting the sentiment of specific news headlines. For fine-tuning, the subset of the Financial Phrase Bank dataset [30] was utilised.

This is separated into 80 percent train, and 20 percent validation datasets. As stated, the goal variable is news sentiment, and the three labels are positive (label 0), negative (label 1), and neutral (label 2). The model is therefore trained, validated, and finally evaluated on the test dataset. The confusion matrix that results is depicted in Figure 18.

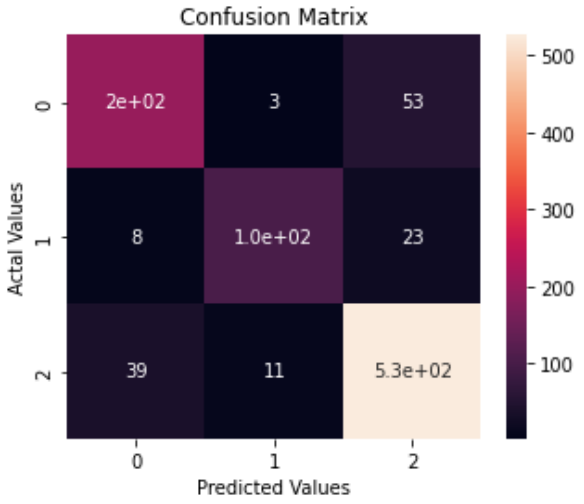


Figure 18 Confusion Matrix

Model	Labels	Precision	Recall	F1-Score	Accuracy
BERT for 1st dataset	Neutral	0.84	0.80	0.82	0.8189
	Positive	0.81	0.83	0.83	
	Negative	0.81	0.81	0.81	
BERT for 2nd dataset	Neutral	0.85	0.81	0.83	0.875
	Positive	0.80	0.87	0.83	
	Negative	0.90	0.91	0.90	

Table 6 Sentiment analysis results

The confusion matrix shown above yields the results as shown in the above table. For 1st dataset of sentiment analysis [30] along with the accuracy, other evaluation metrics such as Precision, Recall, and the weighted average (F1-score) of precision and recall will be assessed. We got 0.8189 accuracies here. For the 2nd dataset of sentiment analysis [31], our model gives nice accuracy of 0.875. We used this trained model in our final web application to get results for unknown headlines.

7.3 Web application user interface

We have created a web application using flask and HTML and interfaced it with our model to get summaries and sentiments for unknown news and their headlines respectively.

The following figure 19 shows us the user interface of our web application. We have created three different web pages to input our unknown news and to show the results from our models.

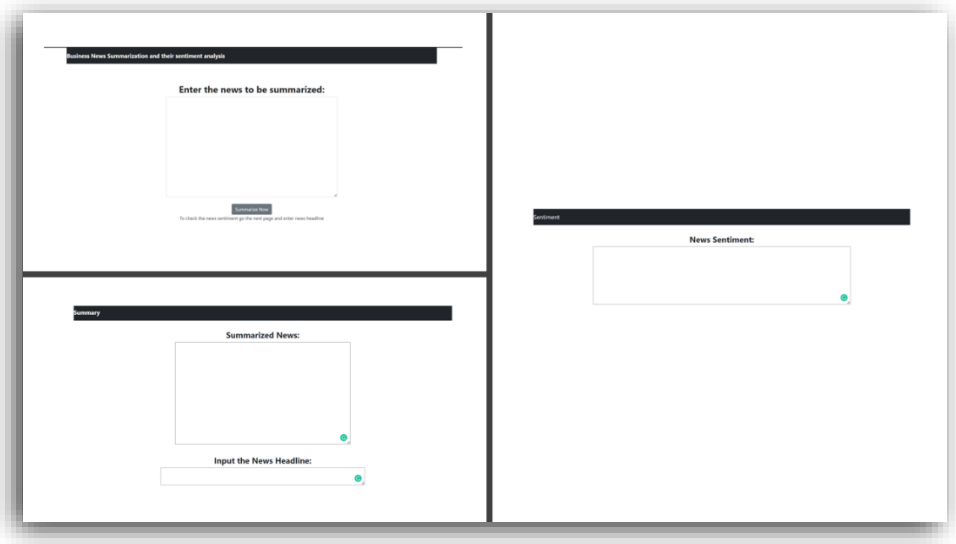


Figure 19 Web Application Interface

Let's see one example of unknown news,

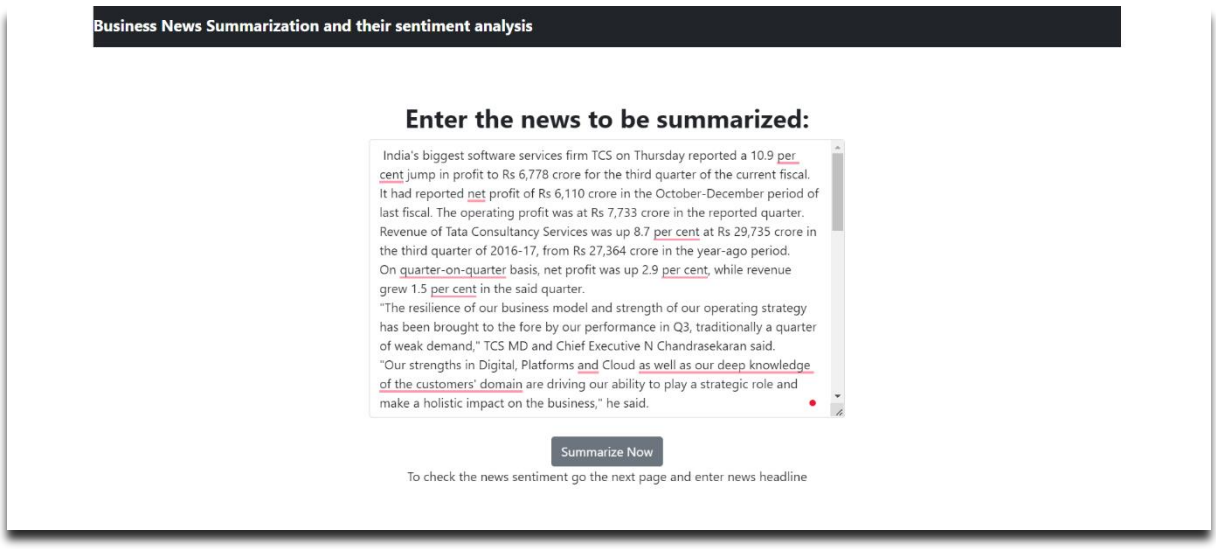


Figure 20 Webpage 1

First, we need to insert the news that we want to be summarized and click on the button given below the box (Summarize Now). It will take nearly 20 to 30 seconds to give us the result.

Figure 21 Webpage 2

The second web page shows us the summarized news as well as it will show us the box where we have to put our news headline to check news sentiment.

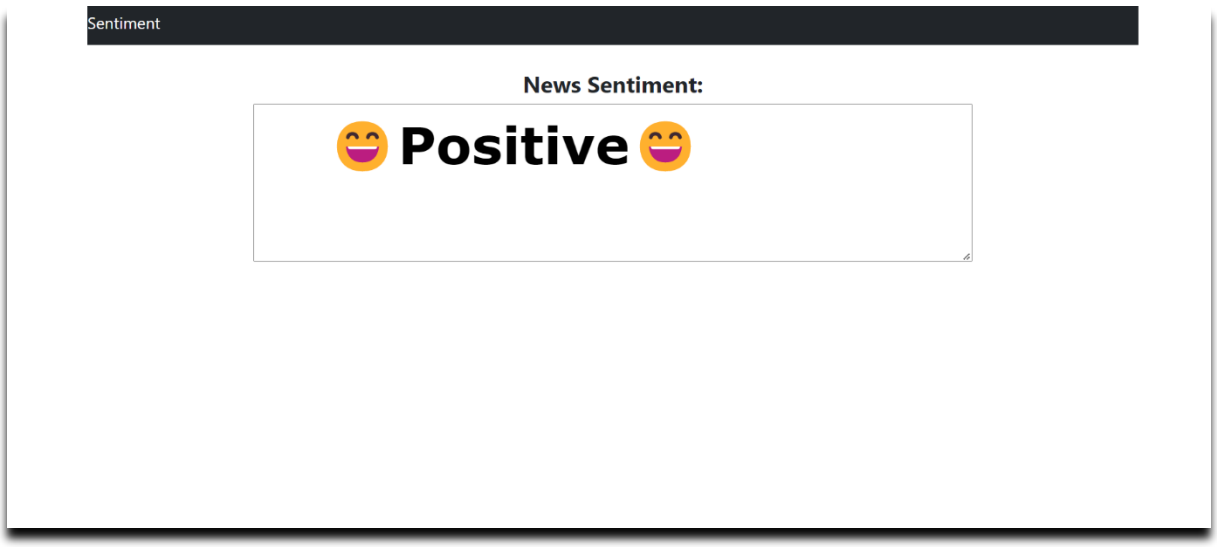


Figure 22 Webpage 3

The above figure shows us the final web page where we get the sentiment from our news headline which we have given.

CHAPTER VIII: Discussions

Overall, the study/project was a success, and the suggested model outperforms the state-of-the-art models for these datasets. We fine-tuned the T5 base model to generate summaries. The BBC News Dataset was used to generate text-based and human-generated summaries. Human-generated summaries are summaries written by humans. The T5 model exhibits favourable outcomes and a relatively higher Rogue score P, R, and F value. The resulting summaries are coherent and accurate. These summaries maintained the text's meaning and were consistent with the original summary.

For news sentiment analysis to predict sentiments of news headlines we used a pre-trained BERT classification model. We used two different datasets to train the BERT model for sentiment analysis. We can use different models like FinBERT, and RoBERT for sentiment analysis they may give some better results on these datasets. Due to time constrain we used the BERT classification model only. We developed a proper end-to-end system which gives us summaries, as well as sentiments of news in one web app, a combination of both tasks, was never seen before.

Though our models performed well there are some limitations as well that we must consider. Like the T5 model takes only 512 token lengths as input and gives a summary of 100 to 150 token length, it creates summaries of fixed length. We are giving news sentiment based on news headlines only, but the news is not just the one factor which affects the stock price there are many other factors as well which must consider. Stock market works on human emotions so we cannot guarantee that if the news sentiment positive stock price will rise.

About future work, we are checking our results only on English news dataset so in future we can consider multilingual summarization and sentiment analysis system. We can build more robust model on regional language business news datasets. In future we can create dataset which contains more business news to increase model accuracy. We are running our web application on local host now so in future we can deploy it on cloud as well. In future, along with sentiment analysis based on news headlines we can combine it with the stock price to get more

CHAPTER IX: Conclusion

In this project, we developed a system which will summarize the business news as well as give its sentiment based on their headline. We have trained and fine-tuned the T5 model and BERT model on our datasets. 'BBC News Summary' is the primary dataset utilised for training our summarization model and for sentiment analysis we primarily used 'FinancialPhraseBank' dataset. To determine our model's applicability for sentiment analysis task we also trained and evaluated it on the 'financial sentiments for Multi-entity News Headlines' dataset.

Based on the results of models on test sets our models achieves good performances as compared to state-of-the-art models. In addition to these tasks to make proper system we created a web application using Flask and hosted it on local server. So that we can get summary and sentiment of unknown news.

The following is a summary of the project's key outcomes:

- A summary of the field of text summarization was provided. Several sorts of summarizing and sentiment analysis methods were investigated in literature review. This provided a perspective on potential implementation strategies for our whole system.
- Models were implemented and integrated with our new website through Flask.
- The model created summaries were compared to summaries written by humans, and the quality of both sets of summaries was measured through Rouge score.
- Finally, all results are showed with example for unknown news through our web application and discussed some limitations and future work.

Therefore, all of the objectives indicated in the introduction chapter have been met. We are planning to broaden the reach of this system based on the system created of this project. First, we will continue our research on the created model to identify new, more effective news summarizer and sentiment analyser modelling methodologies. Second, we will adapt this technique for the summarising of multiple news and multi-lingual news with sentiment analysis on them.

As NLP and related domains advance, more news summarising opportunities emerge. However, there are still numerous obstacles to overcome. The greatest obstacle is the availability of data, particularly outside the news domain, as each new application would require accurate data in its own domain and in the appropriate languages. Distinct types of texts have different structures, and the text's most important information may be in different sections. Evaluation also requires high-quality summaries that are difficult to generate manually.

References:

[1] Christian H, Agus M, Suhartono D (2016) Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TFIDF). ComTech: Computer, Mathematics and Engineering Applications 7:285 .

[2] Babar S, Tech-Cse M, Rit (2013) Text Summarization:An Overview

[3] N. S. Shirwandkar and S. Kulkarni, "Extractive Text Summarization Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697465.

[4] Khan, A., Salim, N., 2014. A review on abstractive summarization. J. Theor. Appl. Inf. Technol. 59, 64–72.

[5] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.

[6] Sara Tarannum, Piyush Sonar, Aashi Agrawal, Krishnai Khairnar, " NLP based Text Summarization Techniques for News Articles: Approaches and Challenges," 2021 (IRJTE),VoIume8.

[7] Gupta, A., Chugh, D. and Katarya, R., 2022. "Automated news summarization using transformers". In *Sustainable Advanced Computing* (pp. 249-259). Springer, Singapore.

[8] Liu, Y. and Lapata, M., 2019. ,"Text summarization with pretrained encoders". *arXiv preprint arXiv:1908.08345*.

[9] P. Vinod, S. Safar, D. Mathew, P. Venugopal, L. M. Joly and J. George, "Fine-tuning the BERTSUMEXT model for Clinical Report Summarization," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-7, doi: 10.1109/INCET49848.2020.9154087.

[10] Chen, L.; Qiao, Z.; Wang, M.; Wang, C.; Du, R.; Stanley, H.E. Prediction of stock market index movement by ten data miningtechniques? Mod. Appl. Sci. 2009, 3, 28–42. 8.

[11] Chakraborty, P.; Pria, U.S.; Rony, M.R.A.H.; Majumdar, M.A. Predicting stock movement using sentiment analysis of Twitter feed. In Proceedings of the 2017 6th International Conference on Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT), Himeji, Japan, 1–3 September 2017; pp. 1–6.

[12] Cambria, E., Das, D., Bandyopadhyay, S. and Feraco, A., 2017. Affective computing and sentiment analysis. In *A practical guide to sentiment analysis* (pp. 1-10). Springer, Cham.

[13] Xing, F.Z., Cambria, E. and Welsch, R.E., 2018. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, *50*(1), pp.49-73.

[14] Kalyani, J., Bharathi, P. and Jyothi, P., 2016. Stock trend prediction using news sentiment analysis. arXiv preprint arXiv:1607.01958.

[15] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, p. e1253, 2018.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997. [7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.

[18] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” CoRR, vol. abs/1910.10683, 2019. [Online]. Available: http://arxiv.org/abs/1910.10683

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

[21] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. 2019. arXiv: 1901.02860 [cs.LG].

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805.

[23] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Moham- mad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. “Google’s Neural Machine Translation System: Bridging the Gap between Hu- man and Machine Translation”. In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144.

[24] A. Warstadt, A. Singh, and S. R. Bowman, “Neural network acceptability judgments,” CoRR, vol. abs/1805.12471, 2018. [Online]. Available: http://arxiv.org/abs/1805.12471

[25] Lin, C.Y., 2004, July. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[26] Papers With Code. Document Summarization on CNN / Daily Mail. https : / / paperswithcode . com / sota / document - summarization-on-cnn-daily-mail. Accessed: 2021-03-18.]

[27] Anon, An introduction to google colab - princeton university. Available at: https://mcgrawect.princeton.edu/guides/Google-Colab-Introduction.pdf [Accessed September 20, 2022].

[28] D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006

[29] Malo, P., Sinha, A., Korhonen, P., Wallenius, J., & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the Association for Information Science and Technology, 65(4), 782-796.

[30] Sinha, A. et al., 2022. sentfin 1.0: entity-aware sentiment analysis for financial news. *Journal of the Association for Information Science and Technology*.

[31] Malo, P., Sinha, A., Korhonen, P., Wallenius, J. and Takala, P., 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, *65*(4), pp.782-796.

[32] S. van der Walt, S. C. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," in Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30, March-April 2011, doi: 10.1109/MCSE.2011.37.

[33] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexan- der M. Rush. "OpenNMT: Open-Source Toolkit for Neural Machine Translation". In: CoRR abs/1701.02810 (2017). arXiv: 1701 . 02810.URL: http://arxiv.org/abs/1701.02810.

[34] *Multilingual BERT snippet*. URL: https : / / peltarion . com / knowledge-center/documentation/modeling- view/build- an- ai- model/ pretrained- snippets/ multilingual- bert-snippet.

[35] dishashree26, 2021. Transfer learning: Pretrained models in Deep Learning. *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model [Accessed September 20, 2022].

[36] Widyassari, A.P., Rustad, S., Shidik, G.F., Noersasongko, E., Syukur, A. and Affandy, A., 2020. Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences*.

[37] Briggs, J., 2021. The ultimate performance metric in NLP. *Medium*. Available at: https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460 [Accessed September 20, 2022].

[38] Paper, D. and Paper, D., 2020. Introduction to scikit-learn. *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python*, pp.1-35.

[39] McKinney, W., 2011. pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, *14*(9), pp.1-9.

[40] Lemenkova, P., 2020. Python Libraries Matplotlib, Seaborn and Pandas for Visualization Geo-spatial Datasets Generated by QGIS. *Analele stiintifice ale Universitatii" Alexandru Ioan Cuza" din Iasi-seria Geografie*, *64*(1), pp.13-32.

[41] Gu, K. and Budhkar, A., 2021, June. A package for learning on tabular and text data with transformers. In *Proceedings of the Third Workshop on Multimodal Artificial Intelligence* (pp. 69-73).

[42] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Appendix

Git Repository link provided by University of Birmingham:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2021/sxs1872/-/tree/main

Executing the code

To execute the code use google colab is required. Also the libraries that are required to be installed: NumPy, pandas, transformer, torch, flask, matplotlib, simpletransformer in order to run all codes.

We need to give the correct path to load the datasets as well as the trained model which we have imported.

To open the web app the link in the second last cell of 'Final_combine.ipynb' need to be accessed.