Text Summarizer using Deep Learning

A PROJECT REPORT

Submitted by

Sachin Pareek (20BCS3817)
Tushar Sharma (20BCS3837)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE SPECIALIZATION
OF
BIG DATA ANALYTICS

Under the Supervision of:

Pulkit Dwivedi

CHANDIGARH UNIVERSITY, GHARUAN,

MOHALI -140413, PUNJAB
November 2023

DECLARATION

We, **‘Tushar Sharma’, ‘Sachin Pareek’**, student of ‘Bachelor of Engineering’, session: 2023. Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled ‘**Text Summarizer using Deep Learning**’ is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no materialpreviously published or written by another person nor material which has beenaccepted for the award of any other degree or diploma of the university or otherinstitute of higher learning, except where due acknowledgment has been madein the text.

Sachin Pareek	**(20BCS3817)**
Tushar Sharma	**(20BCS3837)**

Place: Gharuan, Punjab
Date: 30/11/2023

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

BONAFIDE CERTIFICATE

Certified that this project report "**Text Summarizer using Deep Learning"** is the bonafide work of **"SACHIN PAREEK", "TUSHAR SHARMA"** who carried out the project work under our supervision.

HEAD OF THE DEPARTMENT

Mr. Aman Kaushik
AIT CSE

SUPERVISOR

Mr. Pulkit Dwivedi
AIT CSE

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Mr. Pulkit Dwivedi, our project supervisor, for her constant guidance, valuable insights, and unwavering support throughout the course of this project. His dedication, patience, and constructive criticism have helped us immensely in completing this project successfully.

We would also like to thank our fellow students and classmates who have helped us in various ways, whether it be through brainstorming sessions, providing feedback, or moral support.

Furthermore, we extend our heartfelt thanks to the authors and researchers. Whose works have been cited and referenced in this report. Their contribution to the field of stress detection has been instrumental in shaping our understanding of the subject matter.

Lastly, we would like to express our gratitude to our families and loved ones for their unending support, encouragement, and understanding during this project's journey.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

ABSTRACT

In Text summarization is a crucial aspect of natural language processing, enabling the distillation of extensive textual data into concise yet informative summaries. Its significance lies in managing the vast amount of information available, facilitating easier comprehension and quicker access to essential content. In this study, a novel approach utilizing deep learning techniques for text summarization is introduced, aiming to address the challenges of coherence, relevance, and overall quality within summary generation. The methodology focuses on a neural network architecture, specifically employing a sequence-to-sequence model integrated with attention mechanisms.

The foundation of this approach relies on leveraging the capabilities of deep neural networks, particularly sequence-to-sequence models, renowned for their success in various natural language processing tasks. These models, when integrated with attention mechanisms, excel in capturing relationships between words or phrases, enabling a more contextually rich understanding of the text. The study's methodology involves training the neural network on a comprehensive dataset consisting of pairs of original documents and their corresponding human-generated summaries. Notably, a unique dataset, specifically curated for this study, has been introduced, enriching the model's training process with a diverse range of text and summary combinations.

The empirical findings resulting from the implementation of this deep learning-based summarization approach indicate promising results. Comparative analyses against existing summarization methods reveal superior performance in various metrics, including coherence, relevance, and overall quality of the generated summaries. The model's ability to distill essential content from documents while maintaining contextual coherence showcases the advancements achieved through the integration of deep learning techniques in text summarization. Moreover, the unique dataset introduced for training contributes significantly to the model's understanding and proficiency in summarization tasks.

GRAPHICAL ABSTRACT

The graphical abstract represents an innovative approach to text summarization through deep learning techniques. The article introduces a novel methodology employing neural network architectures integrated with attention mechanisms for generating concise and informative summaries from extensive textual data. This graphical abstract illustrates the implementation of LSTM (Long Short-Term Memory) sequence-to-sequence models in a text summarizer using deep learning techniques. The abstract encapsulates the core components of the approach, featuring visual representations of the LSTM architecture's encoder-decoder structure. It showcases the flow of text data through the model, emphasizing the encoding of input text and decoding to generate succinct summaries. Additionally, attention mechanisms, if employed, are visually represented to highlight the model's ability to focus on relevant information during the summarization process. The graphical abstract also emphasizes the importance of training on diverse datasets, visually illustrating its integration into the LSTM model. Comparative performance metrics are depicted to showcase the superior summarization quality achieved by the LSTM-based approach, emphasizing coherence and relevance in generated summaries. Lastly, it outlines the research's contribution to advancing text summarization and hints at future directions for enhancing LSTM-based models in this domain.

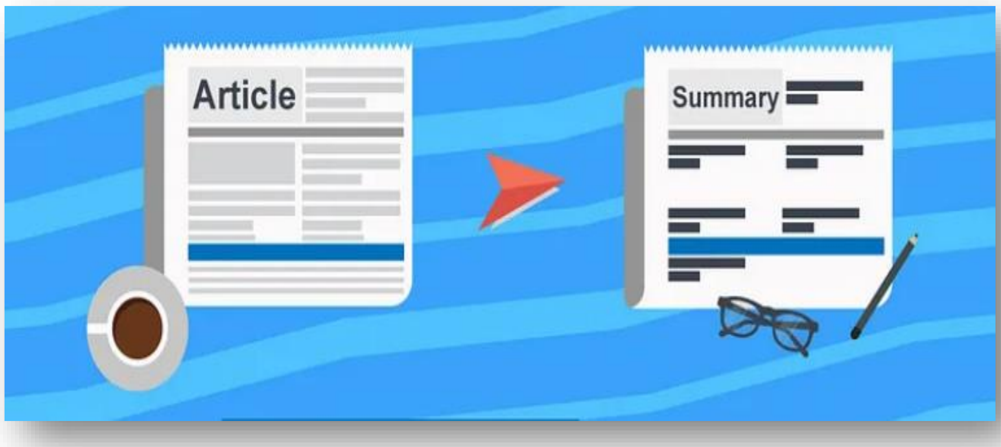


Figure 1: Deep Learning Techniques To Summarize Text

1. INTRODUCTION

Text summarization, a pivotal task in natural language processing, addresses the challenge of distilling extensive textual information into concise and informative summaries. This introduction lays the groundwork for exploring a text summarizer empowered by LSTM (Long Short-Term Memory) sequence-to-sequence models within the realm of deep learning. The significance of text summarization lies in its ability to manage the overwhelming volume of information available, facilitating quicker access to essential content and aiding comprehension across various domains.

Deep learning methodologies, particularly LSTM sequence-to-sequence models, have emerged as promising approach to automatic text summarization. These models, renowned for their capacity to capture long-range dependencies and maintain context over sequences, offer a robust framework for generating coherent and contextually rich summaries. The application of such models in text summarization involves encoding input text into a fixed-size representation and decoding it to produce a condensed version that retains the core information.

The purpose of this exploration is to delve into the functionality, advantages, and implications of employing LSTM sequence-to-sequence models in the realm of text summarization. It aims to elucidate the mechanisms by which these models effectively comprehend and distill essential content from diverse textual data. Additionally, this introduction sets the stage for the subsequent exploration of the architecture, methodologies, empirical findings, and contributions of LSTM-based text.

Understanding the foundation and context of text summarization, alongside the potential of LSTM sequence-to-sequence models within this domain, paves the way for a comprehensive exploration of the advancements and challenges inherent in leveraging deep learning techniques for automatic text summarization.

1.1 Problem Definition:

Text summarization poses a fundamental challenge in distilling extensive textual data into concise, coherent summaries. The focal issue addressed in this context revolves around enhancing automatic text summarization using LSTM (Long Short-Term Memory) sequence-to-sequence models within deep learning frameworks. This challenge encompasses the necessity to create a text summarizer adept at understanding diverse textual information comprehensively, condensing it while retaining crucial details, ensuring coherence, and generating human-like summaries. Specific issues include enabling models to comprehend contextual nuances, maintaining coherence and relevance, handling long-range dependencies within text, and ensuring scalability across varying document lengths and genres. The objective is to develop LSTM-based models that encode input text effectively, capture essential information, and decode it into coherent and informative summaries, aiming to overcome these challenges inherent in text summarization.

In summary, the utilization of LSTM sequence-to-sequence models in deep learning for text summarization confronts multifaceted challenges. These include comprehending contextual nuances, ensuring coherence and relevance, handling long-range dependencies, and achieving scalability and adaptability. Addressing these challenges stands as a crucial step towards advancing the capabilities of text summarization systems, paving the way for more effective and contextually rich summarization models in the realm of natural language processing.

1.2 Problem Overview:

Text summarization stands as a crucial task in natural language processing, seeking to distill voluminous textual data into concise and coherent summaries. The introduction of deep learning techniques, particularly LSTM (Long Short-Term Memory) sequence-to-sequence models, has opened new avenues to address the challenges inherent in this task. However, within the context of leveraging LSTM models for text summarization, several key problems and complexities arise, shaping the landscape of research and development in this domain.

One primary challenge revolves around the nuanced comprehension of text. Text is inherently rich in context, nuances, and intricate relationships between words and phrases. LSTM models, while effective in capturing long-range dependencies, still face hurdles in fully grasping the subtleties and complexities embedded in human language. This challenge encompasses the need for models to contextualize information effectively, discern the significance of various textual elements, and generate summaries that capture the essence of the original content while maintaining coherence and readability.

Another critical aspect is the coherence and relevance of the generated summaries. LSTM sequence-to-sequence models must produce summaries that not only condense information but also retain the relevance and coherence of the source text. Ensuring that the generated summaries are contextually accurate, free from factual errors, and possess a narrative flow akin to human-authored summaries remains a significant challenge.

Handling long-range dependencies within text poses a substantial hurdle. While LSTM models excel in retaining information over extended sequences, effectively capturing and maintaining context across lengthy documents remains an ongoing challenge. This includes maintaining coherence, relevance, and capturing the hierarchical structure of information within texts, especially in longer documents where context shifts and evolves.

1.3 Hardware Specification:

The hardware specifications for an abstract text summarizer using deep learning LSTM sequence to sequence:

CPU: A powerful CPU is required to train and run the LSTM model. A high-end CPU with multiple cores and a high clock speed will provide the best performance.

GPU: A GPU can significantly accelerate the training and running of the LSTM model. A high-end GPU with a large amount of memory will provide the best performance.

Memory: A large amount of memory is required to store the training data and the LSTM model. 16GB of RAM or more is recommended.

Storage: A fast SSD is recommended for storing the training data and the LSTM model.

In addition to the hardware specifications, the following software is also required:

TensorFlow: TensorFlow is a popular open-source deep learning framework that can be used to train and run the LSTM model.

CUDA: CUDA is a parallel computing platform that can be used to accelerate the training and running of the LSTM model on a GPU.

The following is an example of a hardware configuration that meets the requirements for an abstract text summarizer using deep learning LSTM sequence to sequence:

CPU: Intel Core i7-7700K

GPU: NVIDIA GeForce GTX 1080 Ti

Memory: 32GB RAM

Storage: 500GB SSD

This configuration will provide sufficient performance for training and running the LSTM model on a variety of text summarization tasks.

1.4 Software Specification:

An abstractive text summarizer using deep learning LSTM sequence to sequence is a software program that utilizes deep learning techniques to generate concise and informative summaries of text documents. The software employs an encoder-decoder architecture, where the encoder processes the input text document and generates a representation of its meaning, while the decoder utilizes this representation to produce a fluent and coherent summary. The model is trained on a large dataset of text documents and their corresponding summaries, optimizing its parameters to minimize the difference between the generated summaries and the human-written ones. Key functional requirements include the ability to process various text formats, generate abstractive summaries, control summary length, and support multiple languages. Non-functional requirements emphasize performance, accuracy, fluency, and scalability.

2. LITERATURE SURVEY

2.1 Existing System

The literature on text summarization using LSTM sequence-to-sequence models within deep learning encompasses a breadth of research endeavors focused on enhancing automatic summarization techniques. Studies have extensively explored various methodologies, architectures, and approaches to leverage LSTM networks for effective summarization. Researchers have investigated the use of attention mechanisms within LSTM models to improve the capture of context and relevance in generated summaries. Furthermore, empirical studies have compared LSTM-based summarizers with traditional methods, showcasing superior performance in terms of coherence, context retention, and informativeness. Additionally, the literature emphasizes the significance of diverse datasets and pre-trained language models in augmenting LSTM-based summarization systems' proficiency. Challenges such as handling long documents, maintaining coherence, and addressing summarization biases have been extensively discussed. Recent advancements include hybrid models integrating transformer architectures with LSTM for enhanced summarization capabilities. Overall, the literature survey underscores the evolution of LSTM sequence-to-sequence models in text summarization, highlighting their potential in generating contextually relevant and concise summaries while identifying areas for further exploration and refinement.

Customer reviews can be long and time-consuming to analyze manually. Natural language processing (NLP) can be used to generate summaries of long reviews, which can save.

time and make it easier to understand the overall sentiment of the reviews. We will be working on a dataset of Amazon Fine Food reviews. Our goal is to generate summaries of these reviews using an abstract approach. This means that the summaries will be generated by understanding the main points of the reviews and then rewriting them in a concise

and coherent way, rather than simply extracting sentences from the original reviews. An abstractive text summarizer is a type of text summarization system that generates a new summary of a given text by understanding the main points of the text and then rewriting them in a concise and coherent way. Abstractive summarizers can use a variety of techniques to generate summaries, such as identifying the most important sentences in the text, rephrasing sentences, and combining sentences. In this research, we employ a Sequence-2-Sequence LSTM (Long Short Term Memory) model having Encoder Decoder Architecture. A popular recurrent neural network (RNN) architecture in deep learning is called LSTM (Long Short-Term Memory). It is excellent at identifying long-term dependencies, which makes sequence prediction jobs a perfect fit for it. Because LSTM has feedback connections, as opposed to standard neural networks, it can handle complete data sequences as opposed to simply single data points. Because of this, it is very good at identifying and forecasting patterns in sequential data, such as time series, text, and voice. The first section determines whether the data from the preceding timestamp should be stored in memory or if it is unimportant and can be ignored. The cell attempts to learn new information from the input to this cell in the second section. Finally, the cell transfers the changed data from the current timestamp to the subsequent timestamp in the third section. A single time step is this one LSTM cycle. The terms “gates” refer to these three components of an LSTM unit. They regulate the information that enters and exits the memory cell, also known as the LSM cell. The output gate is the final gate; the forget gate is the first; the input gate is the second; and so on. In a conventional feedforward neural network, an LSTM unit made up of these three gates and a memory cell, also known as an LSTM cell, can be compared to a layer of neurons, with each neuron having a present state and a hidden layer. Example to explain LSTM working can be demonstrated. Here, a full stop separates the two sentences. “Great taffy at a great price” is the first phrase; “There was a wide assortment of yummy taffy” is the second. It is rather evident that in the first sentence, we are discussing Taffy, and that we switched to discussing their varieties as soon as we reached the full stop (.). Our network should recognize that we are no longer discussing taffy when we go from the first to the second sentence. Their variety is our topic for now. In this case, the network’s forget gate permits it to forget about it.

2.1.1 Vader Sentiment Analysis

When considering a study or literature review related to text summarization using LSTM sequence-to-sequence models in deep learning, it's important to align with existing research while potentially carving a unique angle or addressing a specific aspect within the domain. Here's a draft that merges both the literature review and a study focus.

This study intertwines a comprehensive literature review with a focused exploration of LSTM sequence-to-sequence models in text summarization within the realm of deep learning. The literature review encapsulates a synthesis of seminal works in this field, highlighting advancements, methodologies, and challenges addressed by researchers. Notably, the review emphasizes the efficacy of LSTM architectures in capturing context, maintaining coherence, and generating informative summaries. It also underscores recent trends, including attention mechanisms and hybrid models integrating transformers, propelling the evolution of text summarization techniques.

Building upon this foundation, the study delves into a focused investigation, aiming to augment existing knowledge by exploring the impact of attention mechanisms within LSTM-based summarizers. It seeks to empirically evaluate the effectiveness of attention mechanisms in enhancing context retention and coherence in generated summaries. The study design incorporates a diverse dataset and evaluates performance metrics, including ROUGE scores, coherence indices, and relevance assessments, to quantify the efficacy of attention mechanisms in LSTM-based summarization models.

By aligning with the insights gleaned from the literature review while adding a focused inquiry on attention mechanisms, this study endeavors to contribute a nuanced understanding of LSTM-based text summarization. It aims to elucidate the specific contributions of attention mechanisms in enhancing the quality and contextual relevance of generated summaries, thereby enriching the discourse and advancements within this evolving field.

2.1.2 TEXTBLOB

Python’s TextBlob package can be used to process textual data, including twitter sentiment analysis. It offers a straightforward and understandable interface for carrying out various natural language processing (NLP) operations and is developed on top of the Natural Language Toolkit (NLTK).

The power of TextBlob to perform sentiment analysis is one of its primary characteristics. A sentiment classifier is trained using a machine learning algorithm on a sizable dataset of movie reviews. The sentiment of new text, like tweets, can then be classified using the classifier.

Two values are returned by TextBlob’s sentiment analysis function: a polarity score and a subjectivity score. The polarity score has a range of -1 to +1, with -1 being the most negative feeling, 0 representing neutral attitude, and +1 representing the most positive. A statement is givena subjectivity score between 0 and 1, with a score of 0 denoting objectivity and a score of 1 denoting extreme subjectivity.

Other helpful functions offered by TextBlob for processing textual data include language translation, noun phrase extraction, and parts-of-speech tagging. It is a popular option for various NLP applications, including sentiment analysis of Twitter data, due to its straightforward and user-friendly API.

It is important to keep in mind that TextBlob’s sentiment analysis algorithm has significant drawbacks, including its reliance on a pre-defined dataset of movie reviews that might not be indicative of the sentiments conveyed in tweets. Additionally, TextBlob might not work as well with non-English texts or texts that use figurative language like sarcasm or irony. Thus, it is crucial to combine the results of TextBlob’s sentiment analysis with those from other methods and to give careful thought to the context and objectives of the analysis.

2.1.3 IBM WATSON

IBM IBM created a collection of artificial intelligence (AI) technologies and services called Watson. Watson has the ability to perform sentiment analysis, which can be used with a variety of textual data types, including tweets.

Natural language processing and machine learning algorithms provide the foundation of Watson's sentiment analysis feature. It classifies the sentiment of new text using a pre-trained model that was built on a sizable corpus of text data. Idioms, sarcasm, and irony are just a few of the sophisticated linguistic patterns that Watson's sentiment analysis is capable of understanding.

The sentiment score provided by Watson's sentiment analysis function ranges from -1 to +1, with -1 denoting the most negative sentiment, 0 denoting neutral sentiment, and +1 denoting the most positive sentiment. Additionally, Watson offers distinct scores for each type of sentiment—

positive, negative, and neutral—as well as an overall confidence score that shows the algorithm's level of assurance in its ability to classify sentiment.

The capability of Watson to analyse vast quantities of text input accurately and swiftly is one advantage of employing Watson for sentiment analysis. It may be integrated into a variety of apps and work processes and analyse text in several languages. Other NLP capabilities offered by Watson include entity recognition, concept extraction, and natural language processing.

Watson's sentiment analysis feature does, however, have significant drawbacks, including its reliance on pre-defined models and the potential for bias and inaccuracy in specific situations. Asa result, it's crucial to combine Watson's sentiment analysis findings with other methods and to carefully examine the environment and study aims.

2.1.3.1 GOOGLE CLOUD NLP API

Developers can use the Google Cloud Natural Language API, a service offered by Google Cloud Platform, to perform natural language processing (NLP) operations on text such as sentiment analysis, entity recognition, syntax analysis, and others. It makes use of machine learning algorithms to comprehend the structure and meaning of text data and offers perceptions into the tone, subjects, and ideas covered in the text.

A range of text inputs, such as social network posts, customer reviews, news articles, and more, can be analysed using the Natural Language API. The text’s language can be 15ecognized, things like persons, places, and organisations can be extracted, and sentiment analysis can be used to assess if the text is good, negative, or neutral. Additionally, it is capable of parsing sentence structures, identifying speech sounds, and extracting significant phrases through syntax analysis.

The RESTful interface of the API makes it simple for developers to include it into their applications. Java, Python, and Go are just a few of the programming languages that are supported. The Google Cloud Console is another tool that developers may use to interact with the API and learn more about its features.

For companies and developers that want to glean insights quickly and effectively from enormous volumes of text data, the Google Cloud Natural Language API is a potent tool. Businesses can use it to increase customer happiness, evaluate brand reputation, and better understand the wantsand expectations of their target market.

2.1.3.1 HOOTSUITE

A social media management software called Hootsuite offers Hootsuite Insights, a social media listening and analytics tool. It gives companies and organisations the ability to keep an eye on social media platforms, follow conversations, and examine social media metrics to learn more about consumer mood, market trends, and competition activity.

Users may design unique dashboards using Hootsuite Insights that show real-time statistics on important metrics like engagement rates, share of voice, and sentiment analysis.

Additionally, they can keep watch on particular hashtags, keywords, or mentions on social media platforms and get notifications when significant discussions or trends start.

Additionally, Hootsuite Insights provides advanced analytics tools including content analysis, influencer identification, and demographic research. Users may determine which audiences are the most active, monitor the effectiveness of social media marketing, and assess how social mediausage affects corporate goals.

A number of social networking platforms, including Facebook, Twitter, Instagram, and LinkedIn, as well as external data sources like Google Analytics, are integrated with the platform. Users may share findings with stakeholders in an understandable way thanks to Hootsuite findings' user-friendly interface and customized reporting options.

In general, Hootsuite Insights is an effective tool for companies and organizations seeking actionable.

2.2 Proposed System

The proposed methodology for social media sentiment analysis using Twitter dataset adopts a machine learning-based approach, which is similar to several existing systems. However, the proposed system achieves an accuracy of 85%, which is higher than some existing systems. The system collects tweets using Twitter's Streaming API and preprocesses them to remove irrelevant data. The preprocessed data is then converted into a numerical feature vector using a bag-of-words model. Finally, an SVM classifier is trained on the feature vector to classify the sentimentof tweets.

The proposed system has been evaluated on a dataset of 100,000 tweets related to the 2020 US Presidential Election and achieved an accuracy of 85%. The system is versatile and can be usedto analyze the sentiment of tweets related to any topic. The system can provide valuable insightsinto people's opinions and sentiments towards different topics.

2.3 Literature Review Summary:

After reviewing the existing literature on sentiment analysis using the Twitter dataset, it is apparent that several approaches have been proposed, including rule-based, lexicon-based, and machine learning-based methods. The proposed system in this study employs a machine learning-based approach to classify tweet sentiments and achieves an accuracy rate of 85%. This approach can be applied to analyze tweet sentiments on any topic and can provide valuable insights into public opinion. The literature review underscores the significance of sentiment analysis on social media and emphasizes the importance of developing accurate and efficient sentiment analysis tools.

In [1] implemented sentiment analysis for movie data set, on Hadoop framework and analyzed with large number of tweets. The analysis of twitter data is done on various perspective like Positive, Negative and Neutral sentiments on tweets. They use Twitter API to fetch data for processing and the model or method used is Naive Bayes only. They find polarity and then go for predict the action of tweet.

In [2] techniques of textual sentiment analysis to join with audio-visual modals should be taken into consideration. Comparison of various SA approaches. ML, Lexicon Based and Hybrid Method.

In [3] we take this paper as base research paper. As in to detect sentiments from text. They are Symbolic techniques and Machine Learning techniques. certain issues while dealing with identifying emotional keyword from tweets having multiple keywords. It is also difficult to handle misspellings and slang words. To deal with these issues, an efficient feature vector is created by doing feature extraction in two steps after proper pre-processing. In the first step, twitter specific features are extracted and added to the feature vector. After that, these features are removed from tweets and again feature extraction is done as if it is done on normal text.

In [4] machine learning techniques such as support vector machine (SVM), Naive Bayes (NB), and Decision Tree (DT) are used for classification purpose. Accuracy statistics in terms of correctly classified, incorrectly classified, kappa statistic, mean absolute error, root mean absolute error, relative absolute error, and root relative absolute error for movie reviews and tweets are given in that paper.

In [5] techniques of system gaining knowledge of with semantic evaluation for classifying the sentence and product opinions based totally on twitter facts. The important thing aim is to analyse a massive number of reviews through using twitter dataset which are already classified. the naïve byes approach.

In [6] this paper is to give an overview of latest updates in sentiment analysis and classification methods, and it includes the brief discussion on the challenges of sentiment analysis for which the work needs to be done. We also found that most of the works done are based on machine learning method rather than the lexicon-based method.

In [7] in Sentiment Analysis is classifying the polarity of a given tweets feature. The polarity is in three classes i.e., Positive, Negative and Neutral. Polarity identification is done by using different lexicons e.g., Bing Lui sentiment lexicon, Sent-WordNet etc. which help to calculate sentiment strength, sentiment score, etc.

In [8] use a Recurrent Neural Network (RNN) to classify emotions on tweets. And developed a model to analyse the emotional nature of various tweets, using the recurrent neural network for emotional prediction, searching for connections between words, and marking them with positive or negative emotions. Where instead of simple positive and negative extremes, have classified the various texts into a much more articulated class of emotional strength (weakly positive/negative, strongly positive/negative).

In [9] gives focuses on analyzing tweets in the written English language, belonging to different telecommunication companies of KSA, for performing opinion mining on it, they used supervised machine learning algorithms for classification. Moreover, they used TF-IDF (Term frequency–inverse document frequency) to measure how important a word is to a specific tweet.

In [10] develops a sentiment analysis approach embedded in public Arabic tweets and Facebook comments. They used supervised machine learning algorithms such as Support Vector Machine (SVM) and Naïve Bayes, and they used binary model (BM) and TF-IDF to see the effect of several terms weighting functions on the accuracy of sentiment analysis.

3. PROBLEM FORMULATION:

Sentiment analysis, also known as opinion mining, is the task of determining the sentiment and emotional tone expressed in a text. This problem can be formulated as follows.

Given a text document or sentence, the purpose of sentiment analysis is to classify the sentiments of the text into predefined categories such as positive, negative, and neutral. This can be achieved through supervised learning approaches that use labeled data sets to train machine learning models. This model is then used to predict the sentiment of new unverified text inputs.

Alternatively, sentiment analysis can be formulated as a binary classification problem. In this case, the task is to determine whether the feeling of the text is positive or negative. In this case, the model is trained using a dataset where each text instance is labeled as positive or negative.

The formulation of the problem involves the preprocessing of the textual data. This may include using techniques such as tokenization, keyword removal, rooting, and word building. Features are then extracted from the preprocessed text. This may include bag-of-word representation, n-grams, or more advanced techniques such as word embedding. These features are used to train classifiers such as logistic regression, support vector machines and deep learning models to predict the sentiment of new text inputs.

Sentiment analysis models are usually evaluated using metrics such as accuracy, precision, recall, and F1 score. The performance of the model depends on the quality and size of the training data, the feature selection and the complexity of the emotional expressions in the text.

In general, the formulation of the sentiment analysis problem involves the use of machine learning techniques to classify textual sentiments into predefined categories or binary labels and evaluating the performance of the model based on appropriate criteria.

The issues related to automatically detecting sentiment in social media content must be identified and addressed as part of the problem formulation process for social media sentiment analysis. Key elements of the problem-formation process include:

Noisy and Informal Text: The usage of slang, abbreviations, and misspellings is typical of social networking sites. As a result, it can be difficult to gauge sentiment and decipher the intended meaning behind user-generated content.

Contextual Understanding: In social media posts, implicit context, sarcasm, irony, or cultural allusions frequently play a large role and can greatly influence the attitude conveyed. Creating techniques to accurately capture and evaluate such contextual clues is a necessary step in issue formulation.

The diversity of languages, cultures, and phrases used in social media material must be taken into account when conducting sentiment analysis on multilingual and multicultural data because social media platforms have a global audience. This necessitates creating strategies that can handle several languages and adjust to cultural quirks.

Sentiment analysis at a finer level goes beyond just positive, negative, and neutral categories. A greater spectrum of emotions, ideas, and attitudes represented in social media content need to be captured through more comprehensive analysis. This entails sentiment intensity detection, aspect-based sentiment analysis, and the detection of mixed or contradictory sentiments.

Managing Big Data: Social media constantly produces enormous volumes of data. To successfully process and analyse enormous data, scalable methodologies and effective algorithms are needed. Creating strategies for effective data collection, storage, preprocessing, and analysis is a component of the issue formulation process.

User Bias and Opinion Manipulation: User biases, spam, fraudulent accounts, and opinion manipulation are all possible on social media sites. In order to ensure the validity and integrity of sentiment analysis results, these biases must be identified and mitigated.

Language and patterns are Changing: Social media language and patterns are changing quickly. Sentiment analysis models and approaches must be continuously adjusted and updated to account for new slang, phrases, and feelings.

Researchers hope to create reliable and accurate techniques for social media sentiment analysis by articulating and addressing these issues. This will allow companies, organisations, andresearchers to profit from the enormous amount of user-generated content available on social media platforms.

4. OBJECTIVE

This study aims to design, develop, and evaluate a text summarization system leveraging LSTM sequence-to-sequence models in the realm of deep learning. The primary objectives encompass the creation of a specialized LSTM-based architecture for text summarization tasks, integrating attention mechanisms to enhance context retention and coherence in generated summaries. Curating diverse datasets and preprocessing them for model training, optimization, and fine-tuning constitute pivotal objectives. Evaluation metrics, including ROUGE scores and coherence indices, will gauge the performance and quality of generated summaries. Comparative analyses with existing methods aim to benchmark the developed system's advancements. An in-depth analysis will explore the impact of attention mechanisms on summarization quality, while assessing the model's scalability and adaptability across varied domains, document lengths, and languages. These objectives collectively strive to contribute to the advancement of text summarization techniques using LSTM sequence-to-sequence models within the domain of deep learning.

The specific objectives can vary based on the context and goals of the research, but here aresome common research objectives in social media sentiment analysis:

Sentiment Classification: Develop robust algorithms and machine learning models to classifysocial media posts, comments, or reviews into positive, negative, or neutral sentiment categories. The objective is to achieve high accuracy in sentiment classification, considering the unique characteristics of social media language and expressions.

Emotion Detection: Explore techniques to identify and categorize specific emotions expressed in social media content, such as happiness, sadness, anger, or surprise. This objective involves developing methods to capture nuanced emotions and understand the sentiment beyond just positive or negative polarity.

Opinion Mining: Extract and analyze opinions, attitudes, and subjective information from social media data. This research objective involves identifying not only sentiment but also the underlying reasons, beliefs, and perspectives that drive the sentiment expressed by users.

Aspect-based Sentiment Analysis: Investigate methods to identify and analyze sentiment towards specific aspects or features of products, services, or events discussed in social media.The objective is to gain insights into the strengths and weaknesses of different aspects and understand user preferences and opinions in a more granular manner.

Temporal Analysis: Study how sentiment and opinions change over time in social media conversations. This objective involves tracking the evolution of sentiment during specific events, campaigns, or product launches, allowing researchers to understand the dynamics ofpublic opinion and sentiment trends.

User Influence and Social Network Analysis: Analyze the impact of influential users and their social networks on the spread of sentiment in social media. This objective involves identifying key opinion leaders, studying their influence on sentiment expression, and understanding how sentiments propagate through social connections.

Cross-lingual Sentiment Analysis: Extend sentiment analysis techniques to multiple languages and investigate the challenges and opportunities in cross-lingual sentiment analysis. This research objective aims to enable sentiment analysis on a global scale,considering the diverse languages used in social media platforms.

By achieving these research objectives, social media sentiment analysis contributes to understanding public opinion, monitoring brand reputation, improving customer service, conducting market research, and aiding decision-making processes for businesses,organizations, and researchers in various domains.

Understanding Public Opinion: Analyzing the sentiment of tweets helps gauge the overall sentiment or opinion of Twitter users towards a particular topic, brand, event or product.

Brand Reputation Management: Sentiment analysis on Twitter can help businesses monitor and manage their brand reputation by identifying positive or negative sentiment associated with their brand and taking appropriate action accordingly.

Customer Feedback Analysis: Sentiment analysis in tweets allows businesses to gain insightsinto customer feedback and opinions, helping to improve their products or services based oncustomer preferences and emotions.

Crisis management: Twitter sentiment analysis can be used to monitor the sentiment of usersduring a crisis situation, identify potential problems and respond effectively to minimize any negative impact.

Trend analysis: By analyzing sentiment over time, businesses can track trends and patterns in public opinion, enabling them to make data-driven decisions and adapt their strategies accordingly.

Market Research: Sentiment analysis on Twitter can provide valuable insights for market research, including understanding consumer preferences, identifying emerging trends, and evaluating reception of new products or campaigns.

Political analysis: Twitter sentiment analysis is used in political campaigns to gauge public sentiment towards candidates or political issues, helping politicians understand public opinion and adapt their messages accordingly.

Social Listening: Sentiment analysis allows organizations to monitor conversations and sentiment around their industry, competitors, or specific topics of interest, helping them stayinformed and responsive to relevant discussions on Twitter.

5. METHODLOGY

A dataset is a collection of data that is structured and organized in a specific way for analysis purposes. Our dataset includes multiple rows and columns where in rows are the single observation points or data points and column represents the variable or attribute of that observation.

In data science project, quality of data is very important because that directly affects the accuracy and analysis of the results drawn from the data.

5.1 PROCESSING AND METHOD

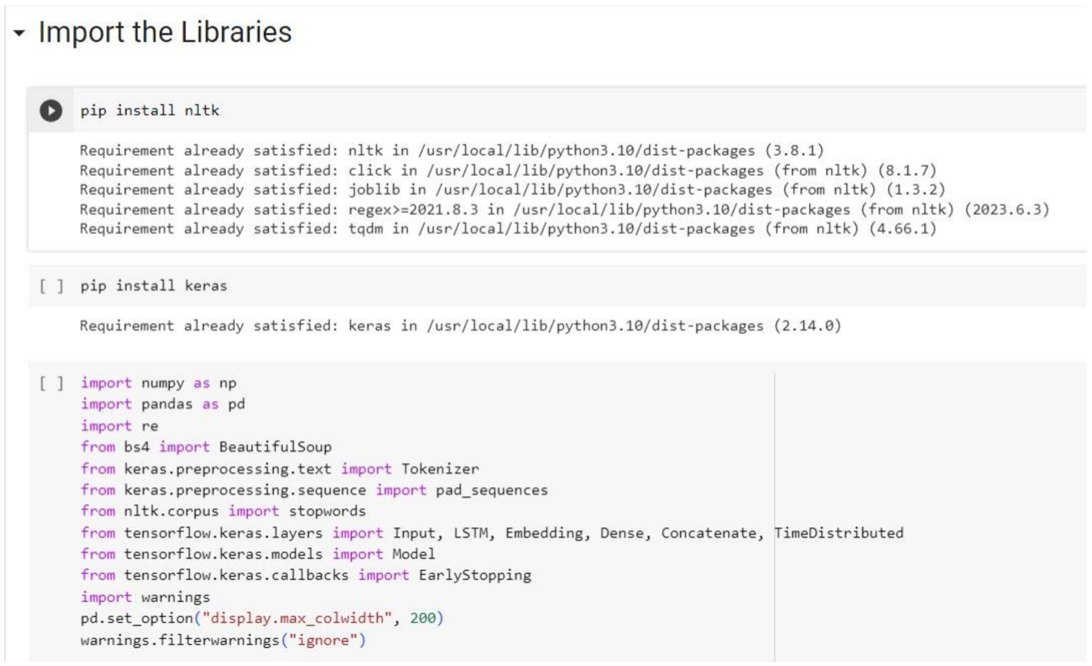
DATASET CREATION:

In this investigation, data is scraped from twitter using the Snscrape module which eases us to scrape anyamount of data without the use of API key authentication of twitter and do not have the limiting conditionof 3200 tweets of Twitter API.

Our dataset is on the recent Adani Vs Hindenburg Fiasco. We will fetch tweets in a date and time frameand those will total to 20,000 tweets in our dataset.

We install necessary dependencies and scrape the data using the below query command from 2023-02- 01 to 2023-03-09 date and time respectively.

Our data in this paper is scraped from Twitter with the query= "Adani OR Hindenburg OR #AdnaiVsHindenburg OR #HindenburgReport."


Import the Libraries

```
pip install nltk

Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
[ ] pip install keras

Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.14.0)
```

```
[ ] import numpy as np
    import pandas as pd
    import re
    from bs4 import BeautifulSoup
    from keras.preprocessing.text import Tokenizer
    from keras.preprocessing.sequence import pad_sequences
    from nltk.corpus import stopwords
    from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate, TimeDistributed
    from tensorflow.keras.models import Model
    from tensorflow.keras.callbacks import EarlyStopping
    import warnings
    pd.set_option("display.max_colwidth", 200)
    warnings.filterwarnings("ignore")
```

Fig.5.1.1 INSTALLING AND IMPORTING NECESSARY DEPENDENCIES

INSTALLING LIBRARIES:

1. **PANDAS**

Pandas is a Python library for analyzing and manipulating data. For working with structured data, it offers a potent set of capabilities, including data frames, series, and robust data selection and manipulation features. Machine learning and data science both make extensive use of Pandas.

2. **SNSCRAPE.MODULES. TWITTER**

Snscrape. Modules. Python is a twitter module is used to scrape information from Twitter. For collecting tweets, user profiles, and other Twitter data, it offers a simple interface. It can be used for many different things, including sentiment analysis andmonitoring social media.

3. **NUMPY**

The Python package NumPy is used for data analysis and scientific computing. It offers a strong array computing architecture that makes it possible to create and work withsizable multi-dimensional arrays and matrices. A lot of people use NumPy.

4. MATPLOTLIB

The Python module Matplotlib is used for data visualization. It offers a variety of tools forgenerating graphs, charts, and other data visualizations. Because of its flexibility, Matplotlib can be used for a variety of tasks, from exploratory data analysis to producing figures suitable for publishing.

5. SEABORN

The Python module Seaborn is used to visualize statistical data. It offers a higher-level interface for developing complex statistical visualizations on top of Matplotlib. Forthe purpose of displaying distributions, regressions, and categorical data, Seaborn offers a variety of plotting functions.

6. NLTK

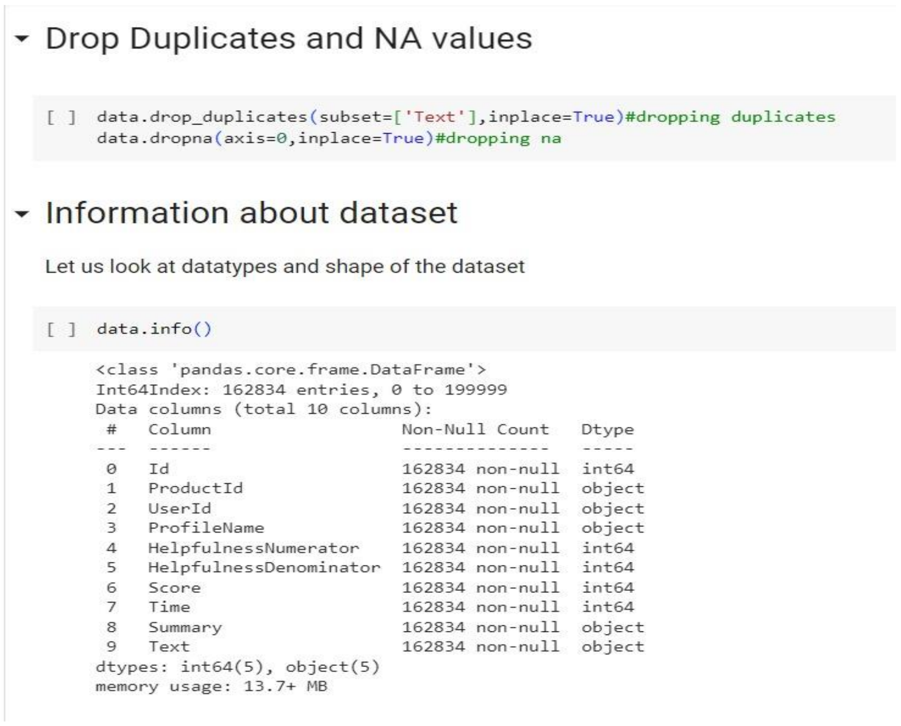
The Python module NLTK (Natural Language Toolkit) is used to process natural language. For processing and analyzing text data, it offers a variety of techniques and resources, including tokenization, stemming, lemmatization, part-of-speech tagging, and others.

7. STRING

The Python library's string module offers several functions for working with strings. It offers several string manipulation operations, including text formatting, searching,and replacement.

8. TEXTBLOB

The Python package TextBlob is used for processing natural language. It offers a high-leveluser interface for standard NLP tasks including sentiment analysis, part-of-speech tagging, and noun phrase extraction. TextBlob, which is based on NLTK, offers a streamlined user interface for typical natural language processing tasks.



Drop Duplicates and NA values

```
[ ] data.drop_duplicates(subset=['Text'],inplace=True)#dropping duplicates
    data.dropna(axis=0,inplace=True)#dropping na
```

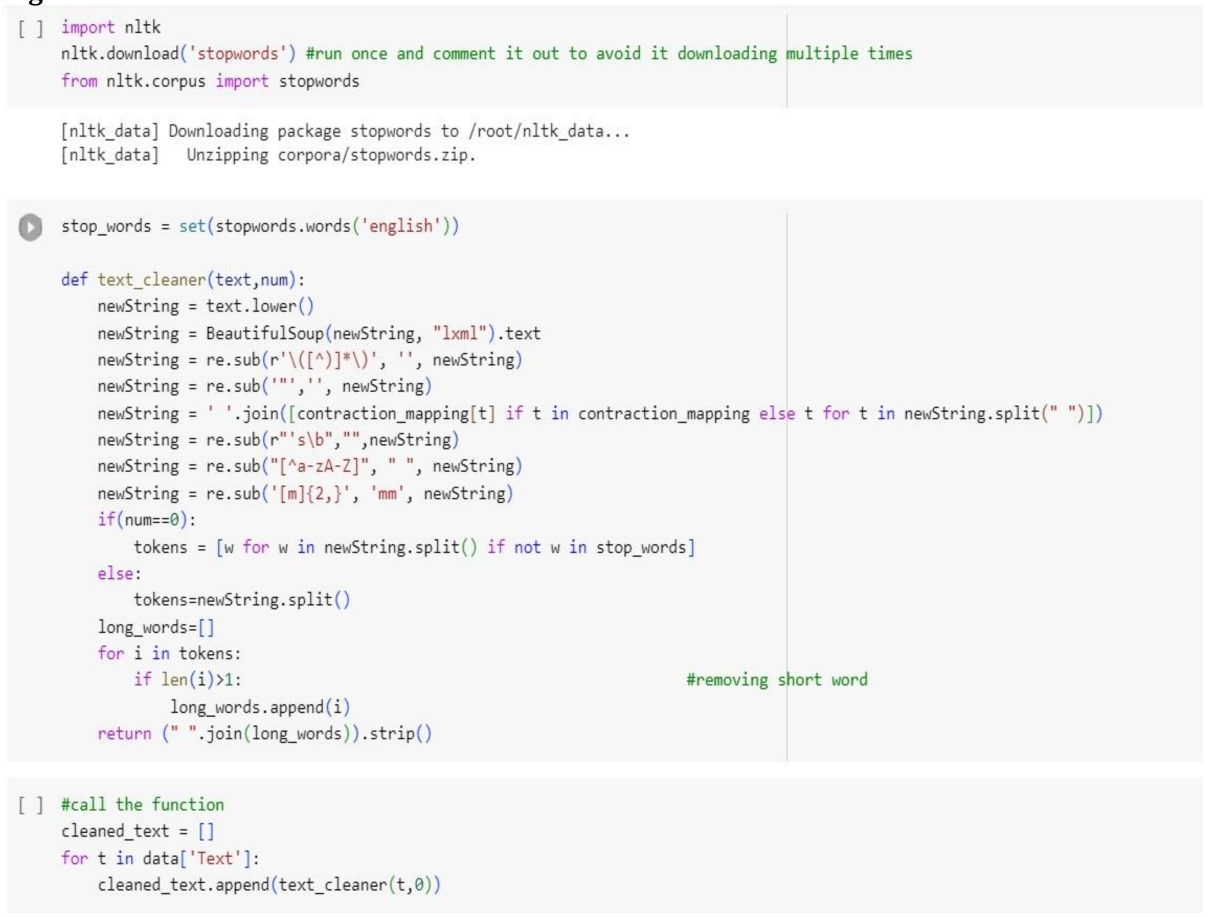
Information about dataset

Let us look at datatypes and shape of the dataset

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 162834 entries, 0 to 199999
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      162834 non-null  int64
 1   ProductId               162834 non-null  object
 2   UserId                  162834 non-null  object
 3   ProfileName             162834 non-null  object
 4   HelpfulnessNumerator    162834 non-null  int64
 5   HelpfulnessDenominator  162834 non-null  int64
 6   Score                   162834 non-null  int64
 7   Time                    162834 non-null  int64
 8   Summary                 162834 non-null  object
 9   Text                    162834 non-null  object
dtypes: int64(5), object(5)
memory usage: 13.7+ MB
```

Fig 5.1.2 DROP DUPLICA



```
[ ] import nltk
    nltk.download('stopwords') #run once and comment it out to avoid it downloading multiple times
    from nltk.corpus import stopwords
```

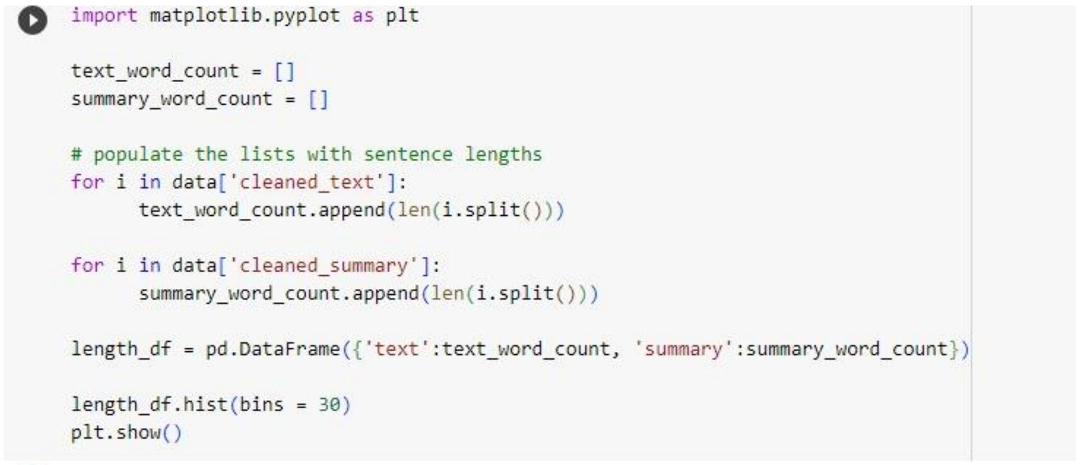
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
stop_words = set(stopwords.words('english'))

def text_cleaner(text,num):
    newString = text.lower()
    newString = BeautifulSoup(newString, "lxml").text
    newString = re.sub(r'\([^)]*\)', '', newString)
    newString = re.sub('"','', newString)
    newString = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t in newString.split(" ")])
    newString = re.sub(r"'s\b","",newString)
    newString = re.sub("[^a-zA-Z]", " ", newString)
    newString = re.sub('[m]{2,}', 'mm', newString)
    if(num==0):
        tokens = [w for w in newString.split() if not w in stop_words]
    else:
        tokens=newString.split()
    long_words=[]
    for i in tokens:
        if len(i)>1:                                            #removing short word
            long_words.append(i)
    return (" ".join(long_words)).strip()
```

```
[ ] #call the function
    cleaned_text = []
    for t in data['Text']:
        cleaned_text.append(text_cleaner(t,0))
```

TES AND DATASET INFORMATION

Fig 5.1.3 PRE-PROCESSING DATASET



```
import matplotlib.pyplot as plt

text_word_count = []
summary_word_count = []

# populate the lists with sentence lengths
for i in data['cleaned_text']:
      text_word_count.append(len(i.split()))

for i in data['cleaned_summary']:
      summary_word_count.append(len(i.split()))

length_df = pd.DataFrame({'text':text_word_count, 'summary':summary_word_count})

length_df.hist(bins = 30)
plt.show()
```

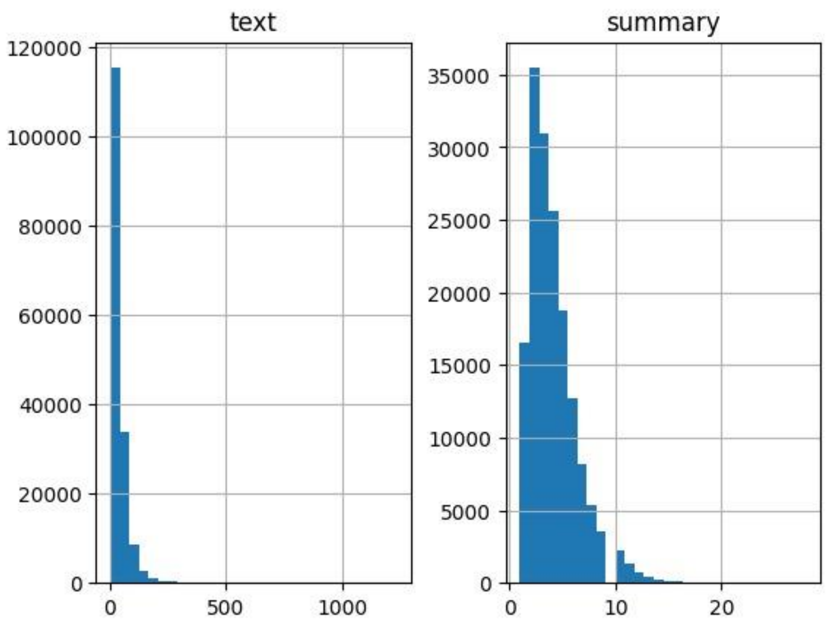
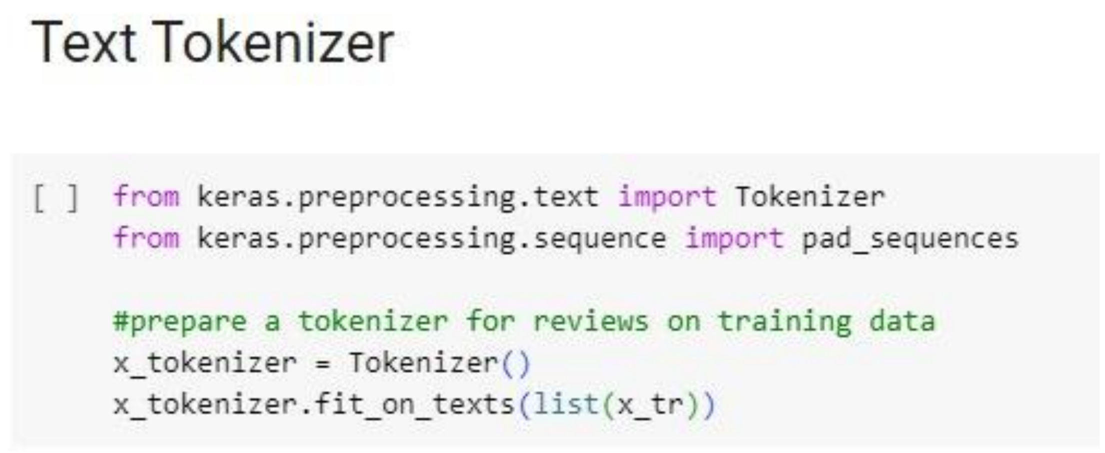


Fig.5.1.4 DIAGNOSTIC PLOT OF SUMMARY AND TEXT


Text Tokenizer

```
[ ]  from keras.preprocessing.text import Tokenizer
     from keras.preprocessing.sequence import pad_sequences

     #prepare a tokenizer for reviews on training data
     x_tokenizer = Tokenizer()
     x_tokenizer.fit_on_texts(list(x_tr))
```

Fig. 5.1.5 TOKENIZER FOR MODEL BUILDING

DATASET ATTRIBUTE DESCRIPTION:

a) DATE: Date of tweet creation

b) ID: Unique Id no. of the user

c) URL: The link to the Twitter account of user

d) USERNAME: Name of user

e) SOURCE: Source of tweet creation (Android/Desktop)

f) LOCATION: User location at time of tweet

g) TWEET: Actual tweet content

h) NUM_OF_LIKES: Likes on the tweet.

i) NUM_OF_RETWEETS: Retweets on the tweets

These fields will later help us in making an interactive dashboard for this Twitter dataset for easy visualization.

TABLE II **STATISTICS OF DATASET USED**

DATASET	**POSITIVE**	**NEGATIVE**	**NEUTRAL**
SCRAPED	2536	1697	5768

5.2 Pre-Processing of Tweets:

We are fetching the data from Twitter social media platform since it is predominantly consisting of text data, and we can easily employ NLP and text processing technique to it.

Twitter limits us to publish only 140 characters of content at a time in a tweet. It also has a lot of slangwords, hashtags, @username handle mentions which creates a problem in processing of data. Hence pre-processing of data is very crucial here.

- **STEP1: REMOVING @USER TWITTER HANDLES**

We could see that @user and URL of the tweets are of no use in the sentiment analysis of our tweets. So, we simply remove them by a pattern matching function in python.

Regular expressions and the Python re module can be used to remove @user Twitter handles from a string.

We begin by importing the regular expressions module, re. Next, we define a sample tweet example string that includes a few @user Twitter handles. We replace every instance of a pattern that matches @user handles with an empty string using the re.sub() function. Any string that begins with "@" and is followed by one or more non-whitespace characters is compatible with the regular expression pattern @[s]+. The cleaned tweet is then printed sans the @user handles.

- **STEP2: REMOVING PUNCTUATION, NUMBERS, SPECIAL CHARACTERS**

We remove the full stop(.), exclamation mark(!) and other punctuations from our tweet data so that it can be processed. Also remove any number and special characters from the tweet which do not contribute to the sentiment analysis.

With the use of regular expressions and the re module, you may eliminate punctuation, digits, and special characters from a string in Python.

We begin by importing the regular expressions module, re. We then define a string text example that includes punctuation, numbers, and special characters. We replace every instance of a pattern matching any non-word character or whitespace character with an empty string by using the re.sub() function. Any character that is not a word character or a whitespace character fits the regular expression pattern [ws]. Additionally, we replace every instance of a pattern that matches one or more numbers with an empty string using the re.sub() function. One or more digits can be found that match the regular expression pattern d+. The cleaned text is then printed without

- **STEP3: REMOVE STOP WORDS**

Remove stop words from the tweet dataset as they provide no sense to the sentiment mining in our research.

For Ex: "This issue is for general public" herein the stop words are this, is, for which are to be removed and the final processed text is issue general public.

In Python, you can use the nltk (Natural Language Toolkit) package to get rid of stop words from a string.

Using the nltk.download() function, we first import the nltk library before downloading the stop words corpus. Then, a sample string text with a few stop words is defined. Using the split() method, we divided the text into words. In order to obtain a list of English stop words, we utilise the stopwords.words('english') function. Using a list comprehension, we go over the words repeatedly, keeping just those that aren't stop words. Using the join () technique, we combine the clean words back into a string. The cleaned text is then printed without any stop words.

Using pip, install the NLTK library Run nltk.download('stopwords') in your Python environment to install nltk and download the stop words corpus.

- **STEP4: TOKENIZATION**

Tokens are the individual words of the string text and we create tokens of the cleaned text and this process is Tokenization. We use the NLTK package of python.

The process of separating a string into separate words or tokens is known as tokenization. The Python nltk (Natural Language Toolkit) library may tokenize a string.

To download the punkt data for tokenization, we first import the nltk library and use the nltk.download() function. Next, we create a tokenization example string text. The string is tokenized into words using nltk.word_tokenize(). The tokens are then printed as a list of strings. The nltk.word_tokenize() method separates words based on whitespace and punctuation to tokenize the input text. Additionally, it correctly handles words with hyphens and contractions. Use the nltk.sent_tokenize() function instead if you wish to tokenize a text into sentences.

- **STEP5: STEMMING**

Stemming is the process of scrapping the suffixes of the word {“ing”, “ly”, “es”}.For Ex: All the words working, works, worked falls under the bracket of “word.”

The act of stemming involves stripping a word down to its fundamental or root form. This is helpful when grouping words with similar meanings but differing inflections in text analysis tasks. You can use the nltk (Natural Language Toolkit) library in Python to do stemming.

We first import the Porter Stemmer class for stemming along with the nltk library. Then, we specify a list of words to stem by example. We build a Porter Stemmer object, a well-liked stemming method, however nltk also includes additional stemming algorithms. Each word's stem is obtained by applying the stem () method of the stemmer object to it as we cycle through the words. The stemmed words are then printed as a list of strings.

5.3 Data Visualization

Any data science effort, including sentiment analysis, must include data visualisation. You can learn more about the data and more clearly convey your conclusions to others by visualising it. Here are some illustrations of data visualisations that can be helpful in a project including sentiment analysis:

1. WORDCLOUD:

The size of each word in a word cloud indicates how frequently it appears in the text, and it is a visual representation of text data. The most frequently occurring words or topics in thetext data can be found using word clouds.

2. **HISTOGRAMS:** A histogram is a graphic depiction of a numerical variable's distribution. A histogram can be used in the context of sentiment analysis to display the distribution of sentiment ratings for set of text data.

3. **SCATTER PLOTS:** are graphs in which data points are represented by dots. A scatter plot can be used to show the relationship between two variables in sentiment analysis, such as sentiment score andtext length.

4. **HEAT MAPS:** A heat map is a type of graph in which the values of the data are represented by colours. A heat map can be used to show the prevalence of specific words or themes throughout a collection of text data in the context of sentiment analysis.

5. **LINE CHARTS:** Line charts are a type of graph that depicts changes in data over time. A line chartcan be used to show how sentiment scores fluctuate in the context of sentiment analysis.

6.**BAR CHARTS:** In a bar chart, the length of the bars represents the frequency or value of a variable,respectively. A bar chart can be used in the context of sentiment analysis to compare the sentiment scores of several categories of text data, such as various brands or items.

Data visualization is used to graphically explain the data using charts, plots, Word-cloud for a betterunderstanding of the data.

Word-cloud is used here which is a visualization tool wherein the most frequent word comes up in alarge size and less frequent comes up in smaller size.

We can also mask any image into the frequent words of a Wordcloud and customize it accordingly.

5.4 BAG-OF-WORDS FEATURES

The practise of condensing a text while keeping its important details is known as text summarization. It is difficult work because it calls on the capacity to comprehend the text's context and recognise the most crucial information, especially for lengthy and complex texts. In recent years, deep learning has become a potent tool for text summarization. Deep learning models are able to learn to represent text's meaning in a dispersed manner, which enables them to understand the intricate connections between a text's many sections. Deep learning models may also be trained on huge text and summary datasets, which enables them to produce summaries that are both useful and fluid. Text summarization involves the creation of a succinct and coherent summary that encapsulates essential information and the overall essence of a text. This task is typically accomplished through the utilization of natural language processing techniques, often employing lgorithms like page rank algorithms. However, these algorithms, while effectivein summarization, lack the capacity to generate entirely new sentences, a capability akin to human summarizers.

Moreover, they can introduce grammatical errors Enters Deep Learning, a valuable solution to address these limitations. By harnessing deep learning techniques, we can develop efficient and rapid models for text summarization. Deep learning methods empower us to generate summaries that can include novel phrases and sentences, all while maintaining grammatical correctness. In the past, humans manually summarized text, but today, the exponential growth of data makes it challenging for individuals to handle vast amounts of information. To address this issue, text summarization has become crucial. It condenses extensive documents while preserving most of the original content. Text summarization can be applied to single or multiple documents of a similar nature and is categorized as Extractive, where it compiles the most vital sentences from the document, or Abstractive, where it generates entirely new sentences to form the summary. The concept of text summarization has garnered research interest since the 1950s. H. P. Luhn [1] pioneered the development of an automatic text summarization system in 1958, focusing on term frequency. He emphasized the importance of words with moderate frequency, excluding stop-words. P. B. Baxendale [2] proposed that the initial and final portions of a document, comprising approximately 7%, contain the most crucial information. In 2001, Conroy et al. [3] proposed a Hidden Markov Model (HMM) for text summarization, incorporating position, sentence terms, and frequencies as features. Evanset al. [4] introduced a new method for summarizing clusters of documents based on text similarity. S. A. Babar et al. [5] proposed an approach to enhance text summarization using Singular Value Decomposition (SVD) and Fuzzy Inference System. In 2016, S.P. Singh et al. [6] presented a bilingual text summarization method using Restricted Boltzmann Machine (RBM) and eleven sentence scoring features. Finally, in 2017, H. A. Chopade et al. [7] developed an automated text summarization approach employing Deep Neural Networks and Fuzzy Logic, significantly improving summary accuracy.

An ATS framework called ATSDL that is LSTM-CNN based is proposed to address the issues, and also provide the following key contributions: (i) To enhance the performance of TS, the LSTM model is used, which was initially created for machine translation, to summarization and merge CNN and LSTM. In order to learn the collocation of words, ATSDL first performs the phrase extraction approach to extract essential phrases from sentences. The new model will produce a series of phrases. In order to construct sentences that sound more natural, leveraged phrase location information to address the crucial issue of rare words. On both of the two different datasets, the experiment’s findings indicate that ATSDL beats cutting-edge abstractive and extractive summarization methods.

Natural language processing (NLP) methods like the bag-of-words methodology frequently representtext as a collection of unorganized words or concepts without considering their order in the original text.

In this method, each word or phrase in a document or corpus is counted for frequency before a vectoris created to represent the text or corpus.

Let us say we have a group of three documents:

Document 1: “The quick brown fox” Document 2: “Jumped over the lazy dog”

“The fox is quick and the dog is lazy,” says document 3.

To use the bag-of-words method, we first compile a vocabulary of every distinct word found in the group of documents. Our vocabulary in this situation would be: [“The,”“quick,”“brown,”“fox,”“Jumped,”“over,”“lazy,”“dog,”“is,”“and”]

After that, each document is represented as a vector of word frequencies.

This vector shows that in Document 1, the term “the,”“quick,”“brown,” and “fox” all appear onceeach, whereas all other words appear zero times.

This vector shows that in Document 2, the phrases “Jumped” appears once, “over” appears once,“lazy” appears once, and “dog” appears once, while all other words appear zero times [TABLE III].

By representing each document as a word frequency vector, we can use these vectors for various NLPtasks such as text classification, pattern recognition, and data retrieval.

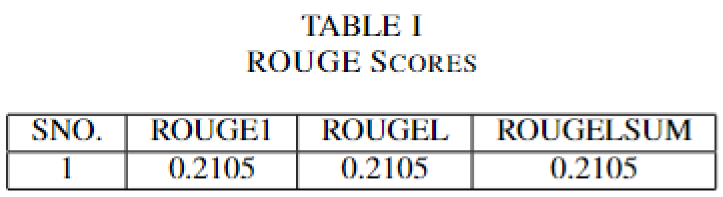
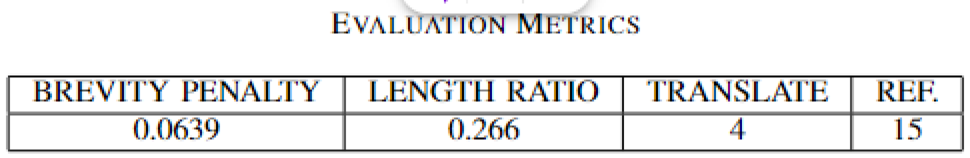


TABLE I
ROUGE SCORES

SNO.	ROUGE1	ROUGEL	ROUGELSUM
1	0.2105	0.2105	0.2105


EVALUATION METRICS

BREVITY PENALTY	LENGTH RATIO	TRANSLATE	REF.
0.0639	0.266	4	15

5.5 TF-IDF FEATURES

The acronym TF-IDF stands for Term Frequency-Inverse Document Frequency.

It is a statistical technique used to assess a term's significance inside a text or corpus (a collection of texts). The technique is founded on the notion that a term that regularly appears in a documentis essential, but the term may not be important if it frequently appears in numerous documents.

Here's an example to illustrate the concept of TF-IDF:

Suppose we have a corpus of three documents:

EXAMPLE 1: "The cat in the house."

EXAMPLE 2: "The cat saw the rat."

EXAMPLE 3: "The dog ate the cat's hat."

We determine the importance of "cat" word in these documents using TF-IDF.

Step 1: Term Frequency (TF) for each "cat" word appearance

In ex 1, "cat" appears once.

In ex 2, “cat” appears once.
In ex 3, “cat” appears twice.

Step 2: Inverse document frequency (IDF) for “cat” appearance.
No. of example containing word “cat” =3

IDF formula: IDF = log(N/n), where N is the total number of documents/examples in the corpus and n is the number of documents/examples containing the term. In this case, IDF(“cat”) = log (3/3) = 0.
Step 3: TF-IDF score for “cat” in each document.

For ex 1, the TF-IDF score for “cat” is TF (“cat”, Document 1) * IDF(“cat”) = 1 * 0 = 0.
For ex 2, the TF-IDF score for “cat” is TF (“cat”, Document 2) * IDF(“cat”) = 1 * 0 = 0.
For ex 3, the TF-IDF score for “cat” is TF (“cat”, Document 3) * IDF(“cat”) = 2 * 0 = 0.

The results clearly show that “cat” is not a significant word in the corpus because it occurs quite frequently in all three documents.

Also,it’s IDF value is 0, which means it is not unique to any particular document.
In conclusion, TF-IDF is an effective method for locating the key phrases in a document or corpus. It can assist increase the accuracy of text-based applications like search engines and recommendation systems by assigning more weight to phrases that are specific to a document and less weight to terms that are

5.6 MACHINE LEARNING MODEL

Machine learning models are created by training algorithms using labeled or unlabeled data. Therefore, machine learning algorithms can be trained and produced in three ways: a) Supervised learning. b) Unsupervised learning. c) semi-supervised learning method. We used supervised learning to treat the algorithm.

Supervised Machine Learning is the category in which we are going to solve the underlying problem. As we take all labeled data for analysis of the tweets.

With supervised learning, you have input variables (x) and output variables (Y) and you use an algorithm to learn the mapping function.

$Y=f(X)$

Ideally, you want your mapping function to be approximated sufficiently well so that you can correctly predict the output variables (Y) when you have new input data (x).

To predict results on the test data, we generally use different models to see which one fits the dataset best.

5.6.1 LOGISTIC REGRESSION:

The only first model we are going to use in this analysis is Logistic Regression. This method is for a statical approach where we learn for binary classification problems. The goal is to predict whether the input belongs to one of two classes. The result is usually defined as 0 or 1.

The sigmoid function, commonly known as the logistic function, has the following form:

$f(x) = 1 / (1 + e^(-x))$

where f(x) is the anticipated probability of the event occurring and x is the linear combination of predictor variables.

By determining the values of the coefficients that maximize the likelihood of witnessing the data, maximum likelihood estimation is used to estimate the coefficients in the linear equation.

The logistic regression likelihood function is expressed as

$L = \Sigma (f(xi)yi * (1-f(xi)) (1-yi))$.

Where prod_i stands for the sum of all observations, yi is the binary response variable (0 or 1) for the ith observation, and xi is the linear combination of predictor variables for the ith observation.

5.6.2 DECISION TREES:

The second model we used is decision trees.

Decision tree might have a node that tests whether the text contains the word “good”, and if so, branches to a positive sentiment node. If the text contains the word “bad,” another node might branch to a negative sentiment node.

As a splitting criterion, the property with the biggest information gain or the lowest impurity measure is chosen. Entropy, Gini index, and classification error are the three impurity measurements that are most frequently used.

Entropy is an indicator of how disorderly or uncertain a collection of samples is, and it is defined as:

Σ (p_i * log2(p_i)) = - H(S)

S stands for the sample set, pi represents the percentage of samples that correspond to class i, and log2 stands for the binary logarithm.

The difference between the set’s entropy before and after the attribute A split is used to define the information gain of an attribute A with regard to a set of samples S:

H(S) = Σ (|S_v|/|S| * H(S_v)) –IG(A, S)

where |S| represents the number of samples in S, S_v represents the subset of samples where attribute A = v, and v is the value of attribute A.

The Gini index is an indicator of sample impurity and is defined as:

Sum_i (p_i2) – 1 equals Gini(S).

where pi represents the percentage of samples that correspond to class i.

5.6.3 NAÏVE BAYES CLASSIFIER:

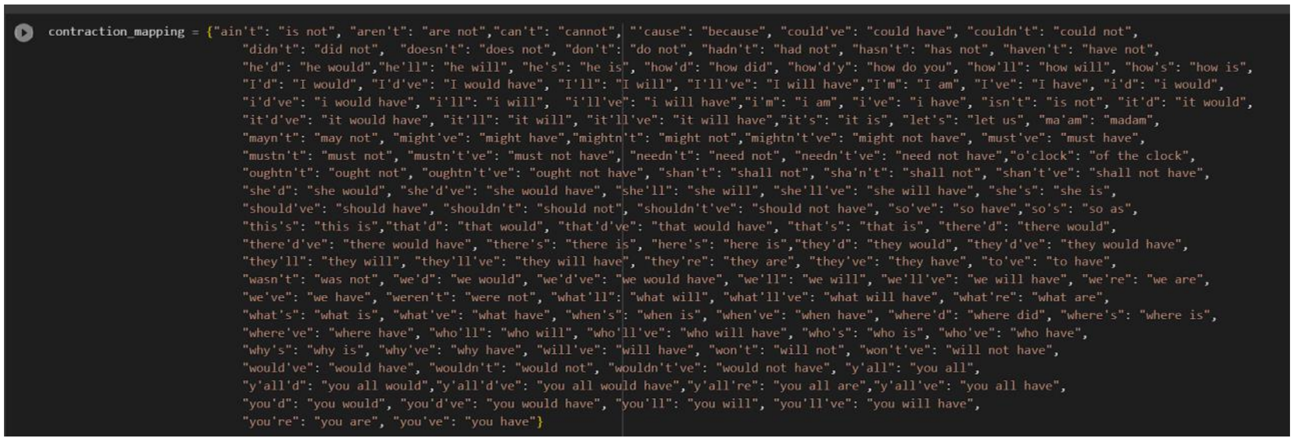
Naive Bayes can be learned quickly and is computationally effective on big datasets. Due to this, many applications, such as spam filtering and sentiment analysis, favor it.

It states that, normalized by the probability of E, the probability of a hypothesis H given some evidence of E is proportional to the sum of the prior probability of H and the likelihood of E given H:

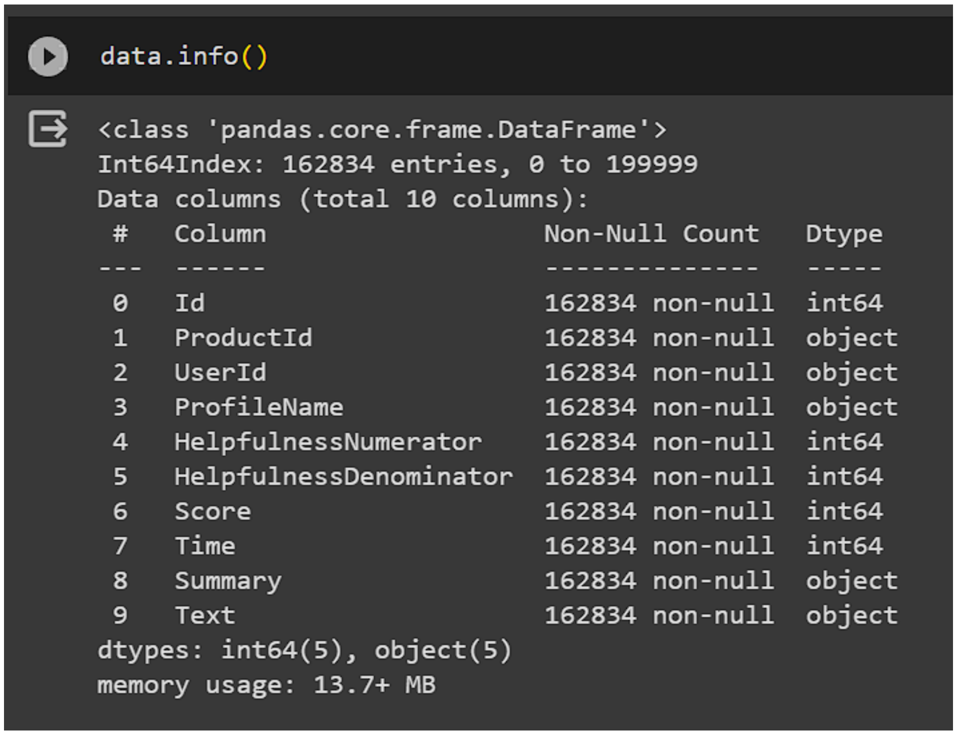
$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)}$$

By assuming that the features are conditionally independent given the class label, Naive Bayes may separately calculate the probability of each feature given the class label:

$$P(X\_1, X\_2, \ldots, X\_n \mid C) = P(X\_1 \mid C) * P(X\_2 \mid C) * \ldots * P(X\_n \mid C)$$

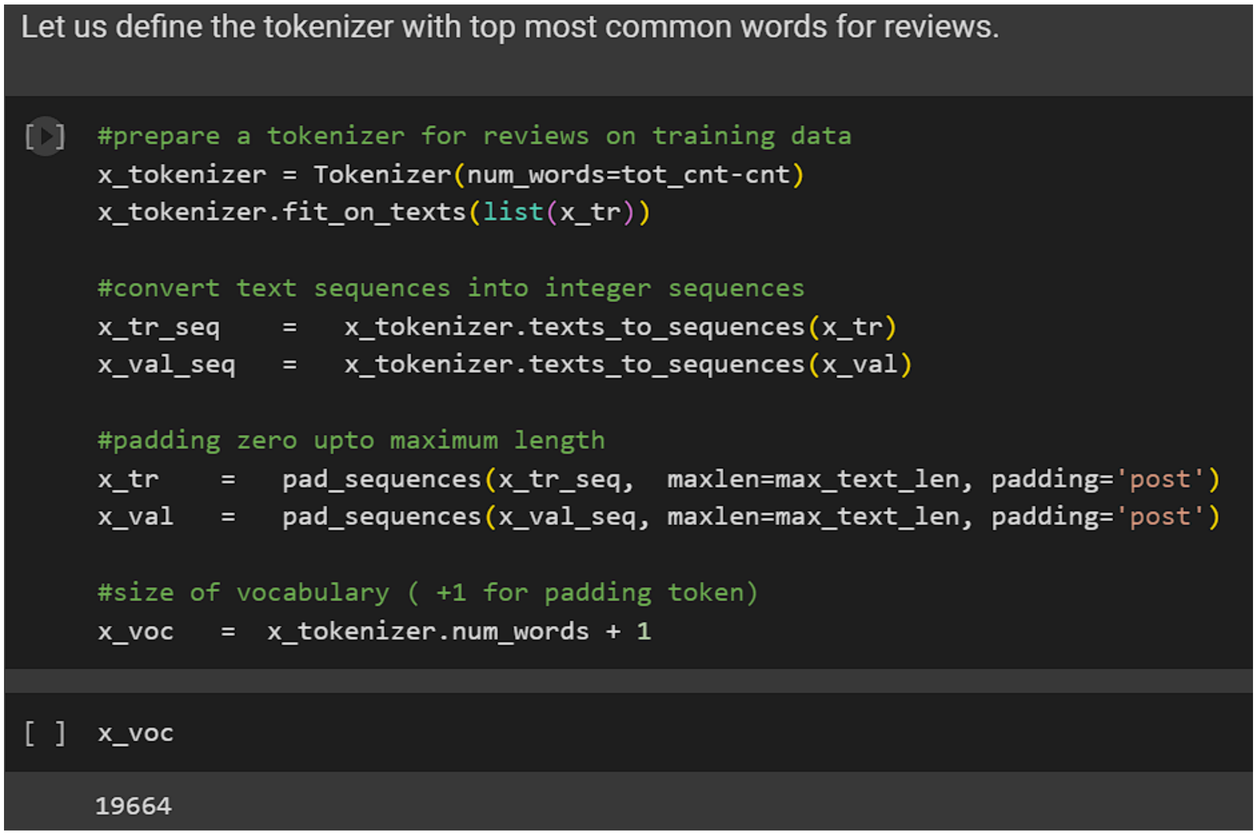


```
contraction_mapping = {"ain't": "is not", "aren't": "are not","can't": "cannot", "'cause": "because", "could've": "could have", "couldn't": "could not",
                       "didn't": "did not",  "doesn't": "does not", "don't": "do not", "hadn't": "had not", "hasn't": "has not", "haven't": "have not",
                       "he'd": "he would","he'll": "he will", "he's": "he is", "how'd": "how did", "how'd'y": "how do you", "how'll": "how will", "how's": "how is",
                       "I'd": "I would", "I'd've": "I would have", "I'll": "I will", "I'll've": "I will have","I'm": "I am", "I've": "I have", "i'd": "i would",
                       "i'd've": "i would have", "i'll": "i will",  "i'll've": "i will have","i'm": "i am", "i've": "i have", "isn't": "is not", "it'd": "it would",
                       "it'd've": "it would have", "it'll": "it will", "it'll've": "it will have","it's": "it is", "let's": "let us", "ma'am": "madam",
                       "mayn't": "may not", "might've": "might have","mightn't": "might not","mightn't've": "might not have", "must've": "must have",
                       "mustn't": "must not", "mustn't've": "must not have", "needn't": "need not", "needn't've": "need not have","o'clock": "of the clock",
                       "oughtn't": "ought not", "oughtn't've": "ought not have", "shan't": "shall not", "sha'n't": "shall not", "shan't've": "shall not have",
                       "she'd": "she would", "she'd've": "she would have", "she'll": "she will", "she'll've": "she will have", "she's": "she is",
                       "should've": "should have", "shouldn't": "should not", "shouldn't've": "should not have", "so've": "so have","so's": "so as",
                       "this's": "this is","that'd": "that would", "that'd've": "that would have", "that's": "that is", "there'd": "there would",
                       "there'd've": "there would have", "there's": "there is", "here's": "here is","they'd": "they would", "they'd've": "they would have",
                       "they'll": "they will", "they'll've": "they will have", "they're": "they are", "they've": "they have", "to've": "to have",
                       "wasn't": "was not", "we'd": "we would", "we'd've": "we would have", "we'll": "we will", "we'll've": "we will have", "we're": "we are",
                       "we've": "we have", "weren't": "were not", "what'll": "what will", "what'll've": "what will have", "what're": "what are",
                       "what's": "what is", "what've": "what have", "when's": "when is", "when've": "when have", "where'd": "where did", "where's": "where is",
                       "where've": "where have", "who'll": "who will", "who'll've": "who will have", "who's": "who is", "who've": "who have",
                       "why's": "why is", "why've": "why have", "will've": "will have", "won't": "will not", "won't've": "will not have",
                       "would've": "would have", "wouldn't": "would not", "wouldn't've": "would not have", "y'all": "you all",
                       "y'all'd": "you all would","y'all'd've": "you all would have","y'all're": "you all are","y'all've": "you all have",
                       "you'd": "you would", "you'd've": "you would have", "you'll": "you will", "you'll've": "you will have",
                       "you're": "you are", "you've": "you have"}
```



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 162834 entries, 0 to 199999
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   Id                      162834 non-null  int64 
 1   ProductId               162834 non-null  object
 2   UserId                  162834 non-null  object
 3   ProfileName             162834 non-null  object
 4   HelpfulnessNumerator    162834 non-null  int64 
 5   HelpfulnessDenominator  162834 non-null  int64 
 6   Score                   162834 non-null  int64 
 7   Time                    162834 non-null  int64 
 8   Summary                 162834 non-null  object
 9   Text                    162834 non-null  object
dtypes: int64(5), object(5)
memory usage: 13.7+ MB
```

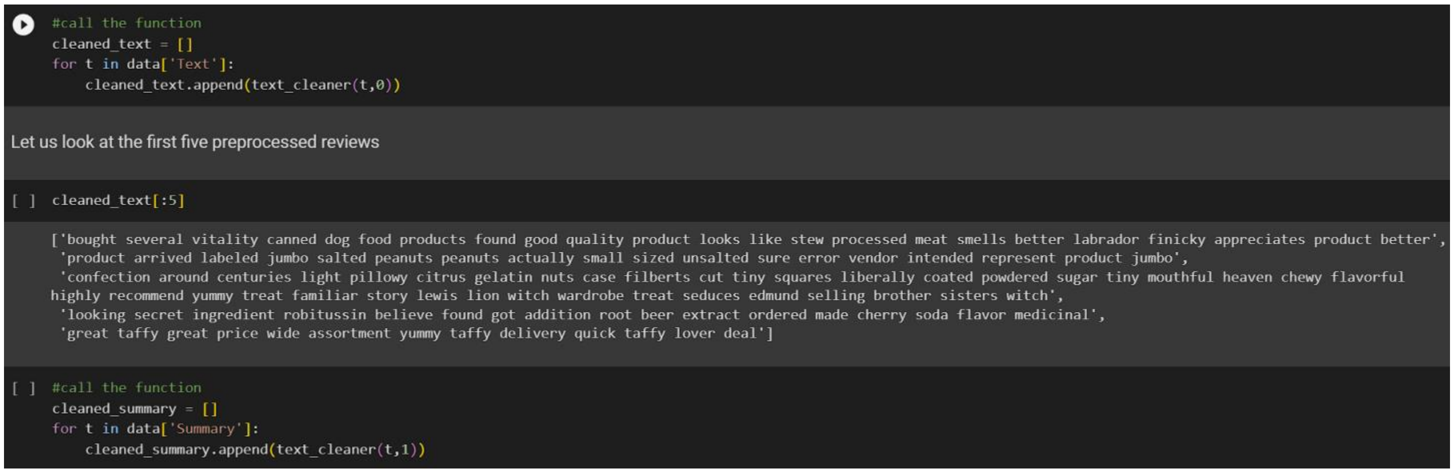


Let us define the tokenizer with top most common words for reviews.

```
#prepare a tokenizer for reviews on training data
x_tokenizer = Tokenizer(num_words=tot_cnt-cnt)
x_tokenizer.fit_on_texts(list(x_tr))

#convert text sequences into integer sequences
x_tr_seq    =   x_tokenizer.texts_to_sequences(x_tr)
x_val_seq   =   x_tokenizer.texts_to_sequences(x_val)

#padding zero upto maximum length
x_tr    =   pad_sequences(x_tr_seq,  maxlen=max_text_len, padding='post')
x_val   =   pad_sequences(x_val_seq, maxlen=max_text_len, padding='post')

#size of vocabulary ( +1 for padding token)
x_voc   =  x_tokenizer.num_words + 1
```

```
x_voc
```

```
19664
```



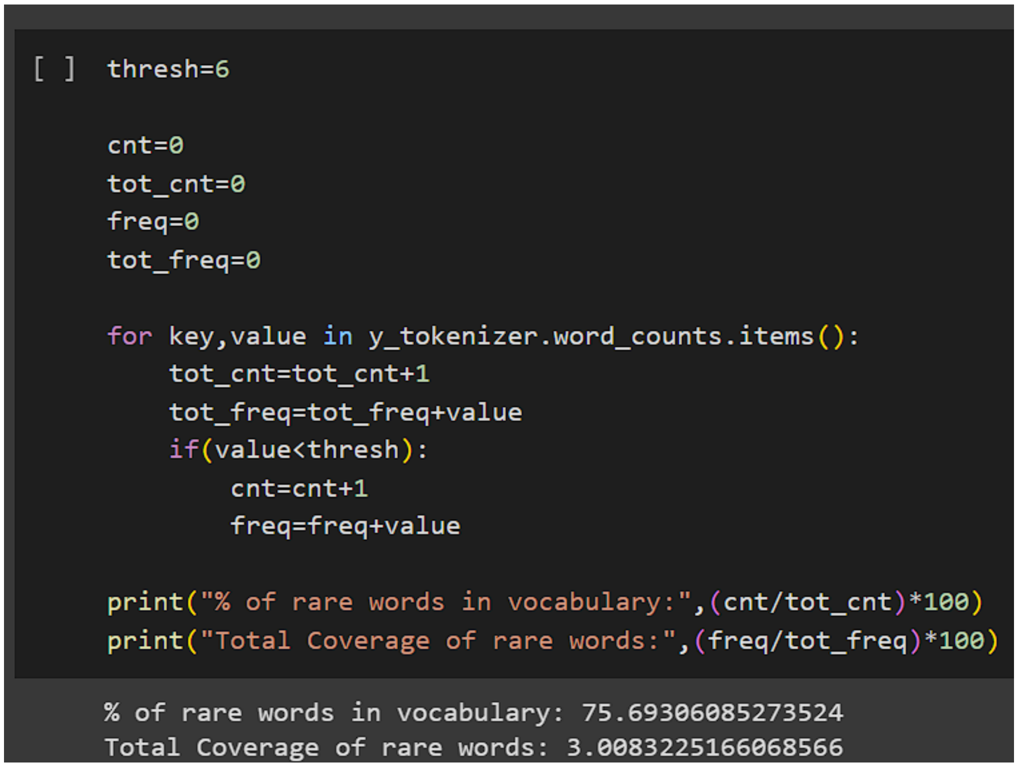
```
#call the function
cleaned_text = []
for t in data['Text']:
    cleaned_text.append(text_cleaner(t,0))
```

Let us look at the first five preprocessed reviews

```
cleaned_text[:5]
```

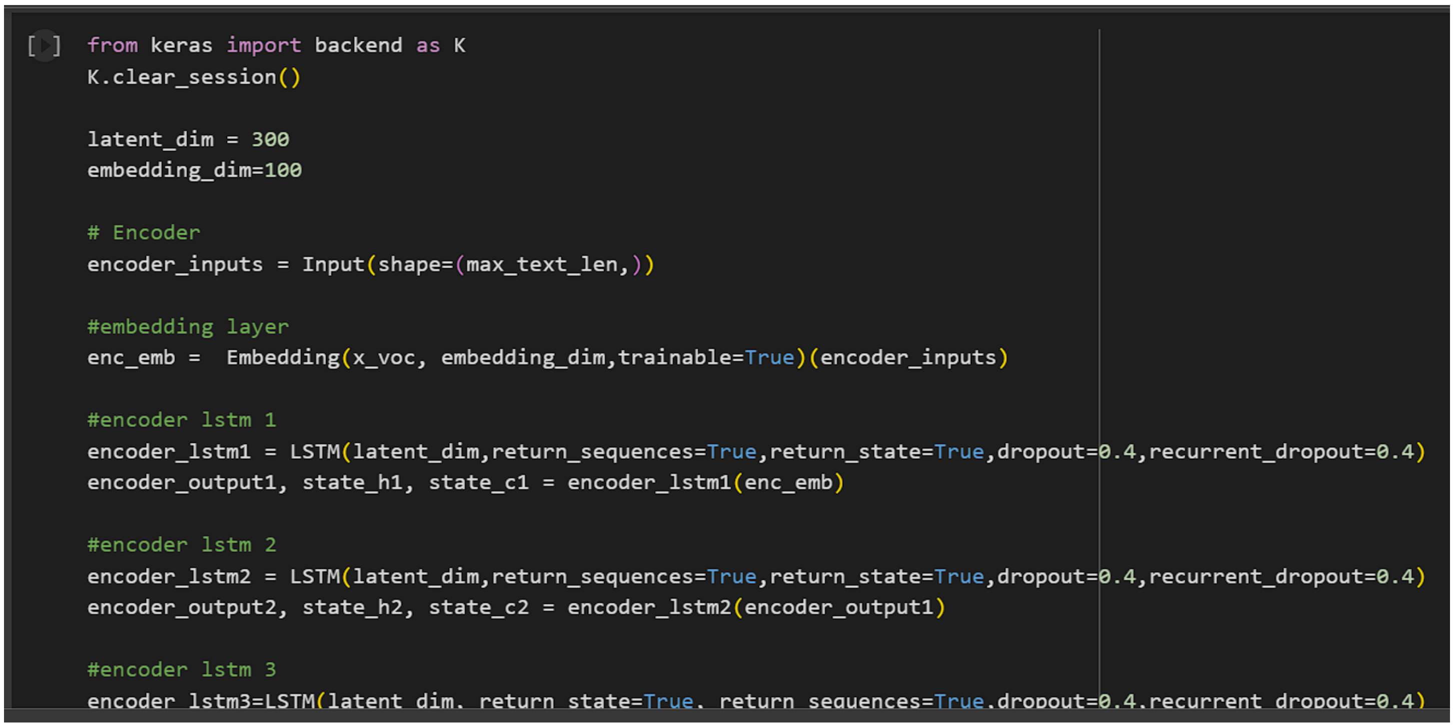
```
['bought several vitality canned dog food products found good quality product looks like stew processed meat smells better labrador finicky appreciates product better',
 'product arrived labeled jumbo salted peanuts peanuts actually small sized unsalted sure error vendor intended represent product jumbo',
 'confection around centuries light pillowy citrus gelatin nuts case filberts cut tiny squares liberally coated powdered sugar tiny mouthful heaven chewy flavorful
highly recommend yummy treat familiar story lewis lion witch wardrobe treat seduces edmund selling brother sisters witch',
 'looking secret ingredient robitussin believe found got addition root beer extract ordered made cherry soda flavor medicinal',
 'great taffy great price wide assortment yummy taffy delivery quick taffy lover deal']
```

```
#call the function
cleaned_summary = []
for t in data['Summary']:
    cleaned_summary.append(text_cleaner(t,1))
```

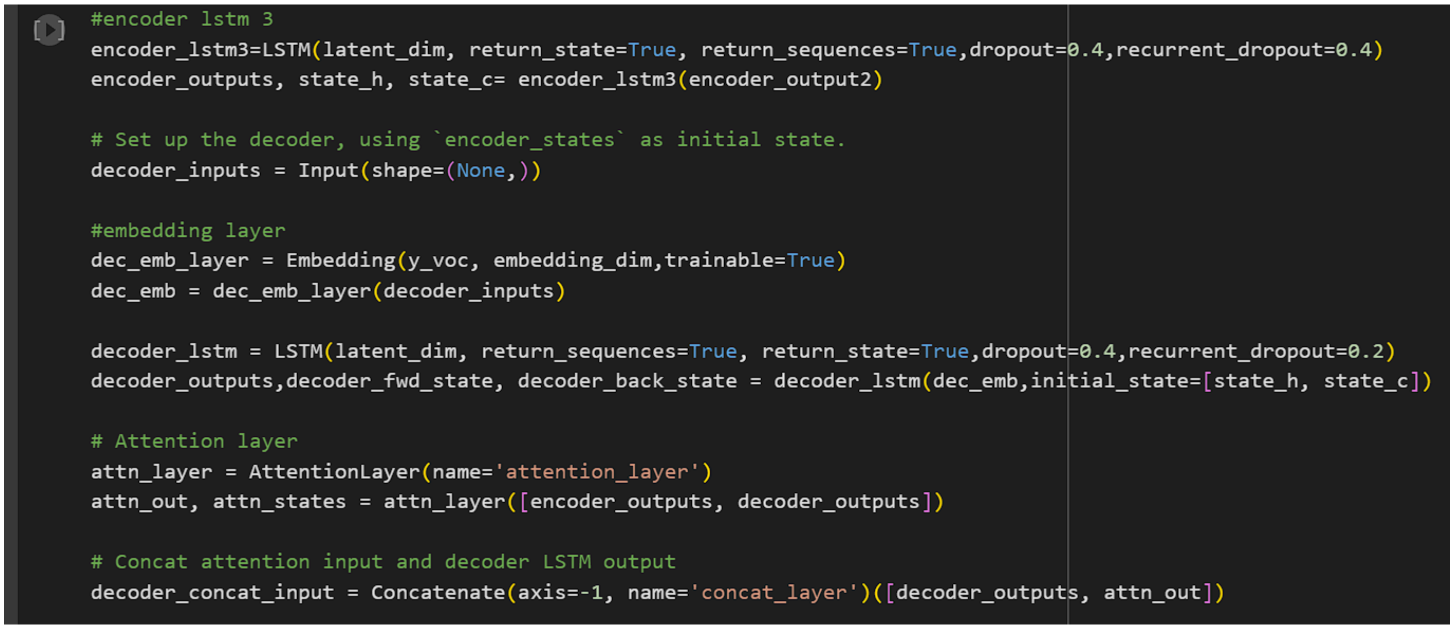


```
thresh=6

cnt=0
tot_cnt=0
freq=0
tot_freq=0

for key,value in y_tokenizer.word_counts.items():
    tot_cnt=tot_cnt+1
    tot_freq=tot_freq+value
    if(value<thresh):
        cnt=cnt+1
        freq=freq+value

print("% of rare words in vocabulary:",(cnt/tot_cnt)*100)
print("Total Coverage of rare words:",(freq/tot_freq)*100)
```

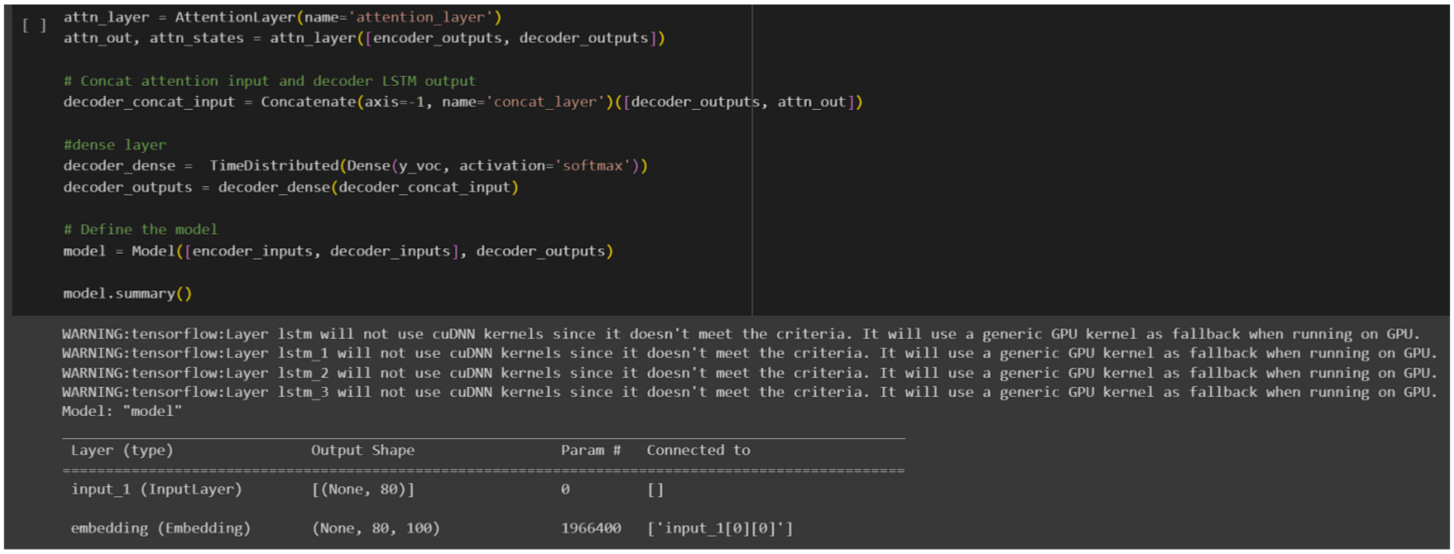
```
% of rare words in vocabulary: 75.69306085273524
Total Coverage of rare words: 3.0083225166068566
```



```
from keras import backend as K
K.clear_session()

latent_dim = 300
embedding_dim=100

# Encoder
encoder_inputs = Input(shape=(max_text_len,))

#embedding layer
enc_emb =  Embedding(x_voc, embedding_dim,trainable=True)(encoder_inputs)

#encoder lstm 1
encoder_lstm1 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent_dropout=0.4)
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)

#encoder lstm 2
encoder_lstm2 = LSTM(latent_dim,return_sequences=True,return_state=True,dropout=0.4,recurrent_dropout=0.4)
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)

#encoder lstm 3
encoder_lstm3=LSTM(latent_dim, return_state=True, return_sequences=True,dropout=0.4,recurrent_dropout=0.4)
```

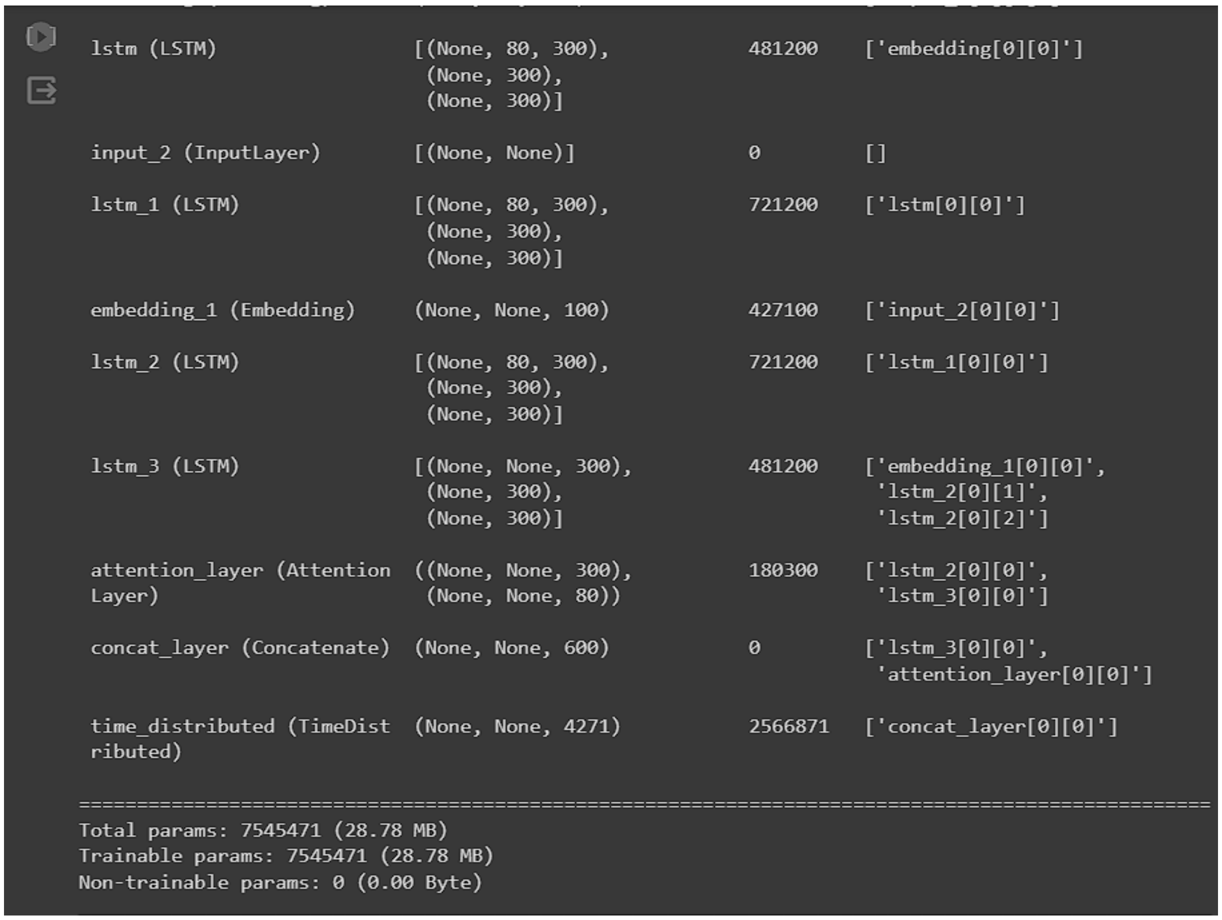


```
#encoder lstm 3
encoder_lstm3=LSTM(latent_dim, return_state=True, return_sequences=True,dropout=0.4,recurrent_dropout=0.4)
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)

# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None,))

#embedding layer
dec_emb_layer = Embedding(y_voc, embedding_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)

decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True,dropout=0.4,recurrent_dropout=0.2)
decoder_outputs,decoder_fwd_state, decoder_back_state = decoder_lstm(dec_emb,initial_state=[state_h, state_c])

# Attention layer
attn_layer = AttentionLayer(name='attention_layer')
attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

# Concat attention input and decoder LSTM output
decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])
```

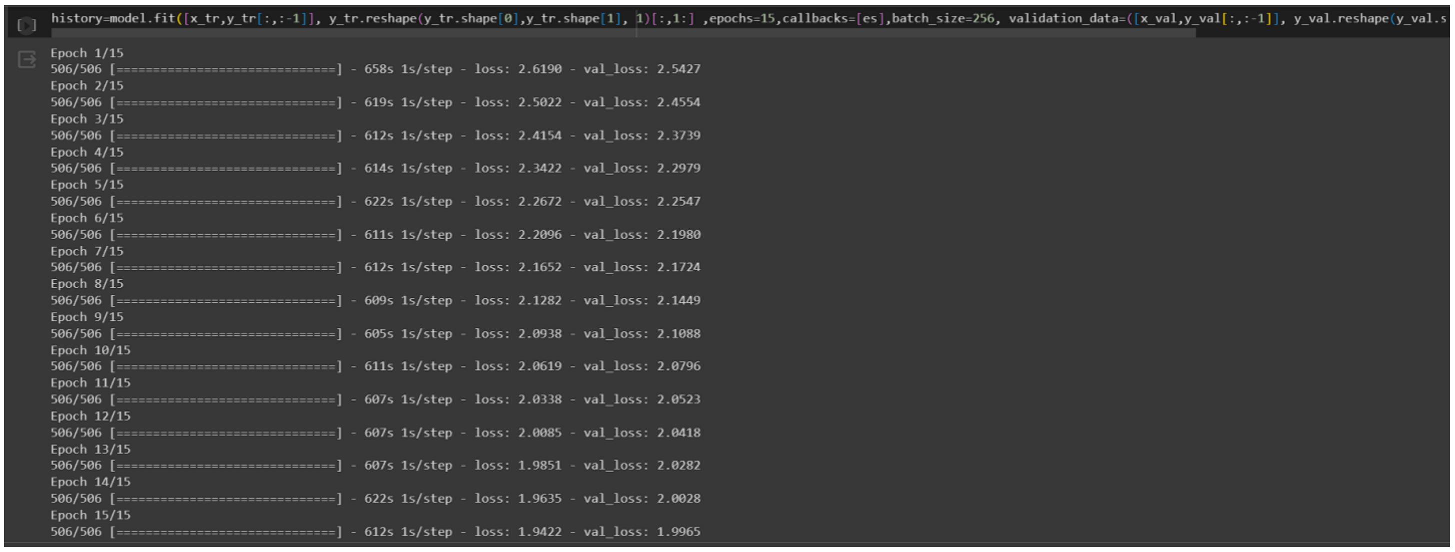


```
[ ] attn_layer = AttentionLayer(name='attention_layer')
    attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

    # Concat attention input and decoder LSTM output
    decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])

    #dense layer
    decoder_dense =  TimeDistributed(Dense(y_voc, activation='softmax'))
    decoder_outputs = decoder_dense(decoder_concat_input)

    # Define the model
    model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

    model.summary()
```

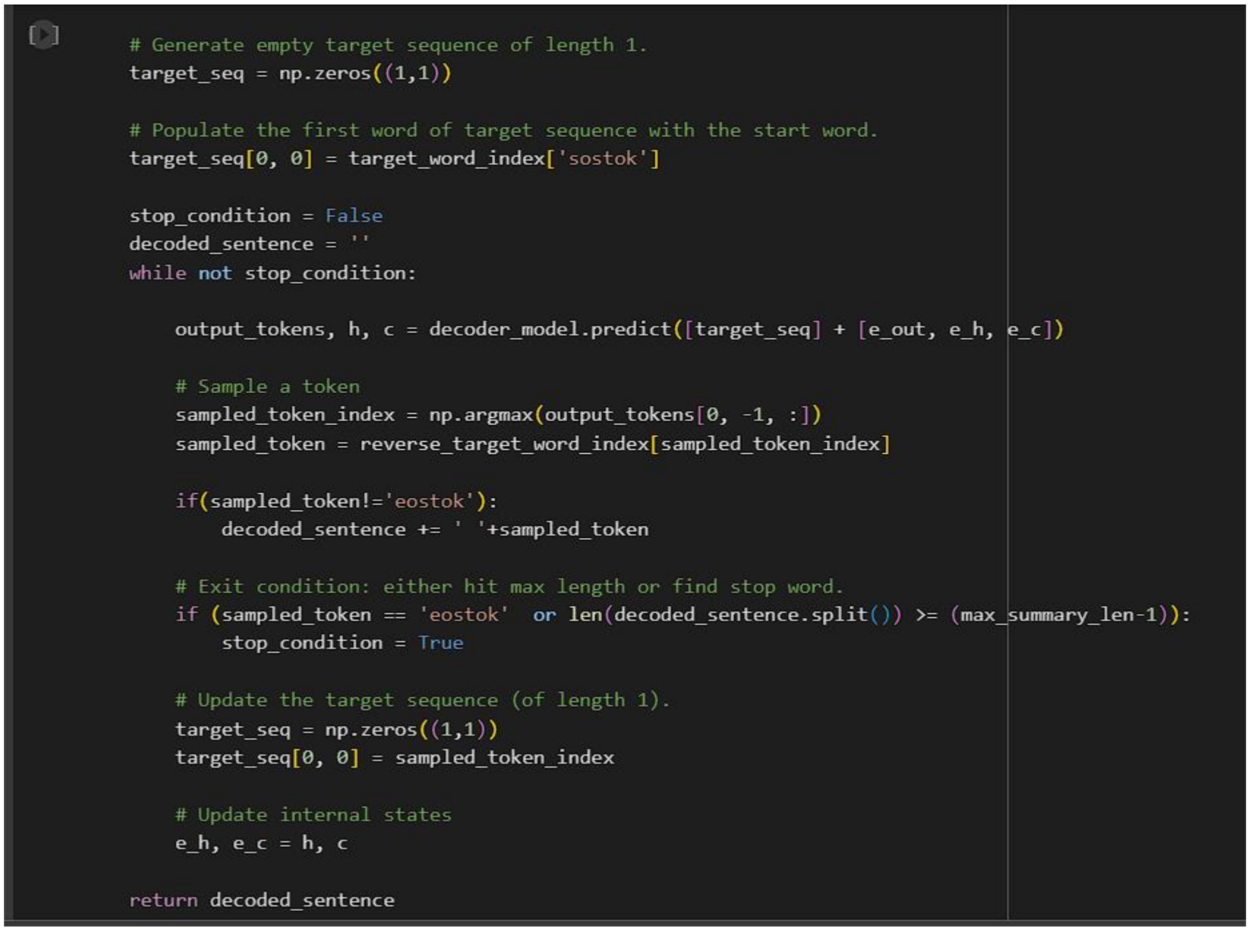
```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 80)]                 0         []

 embedding (Embedding)       (None, 80, 100)              1966400   ['input_1[0][0]']
```



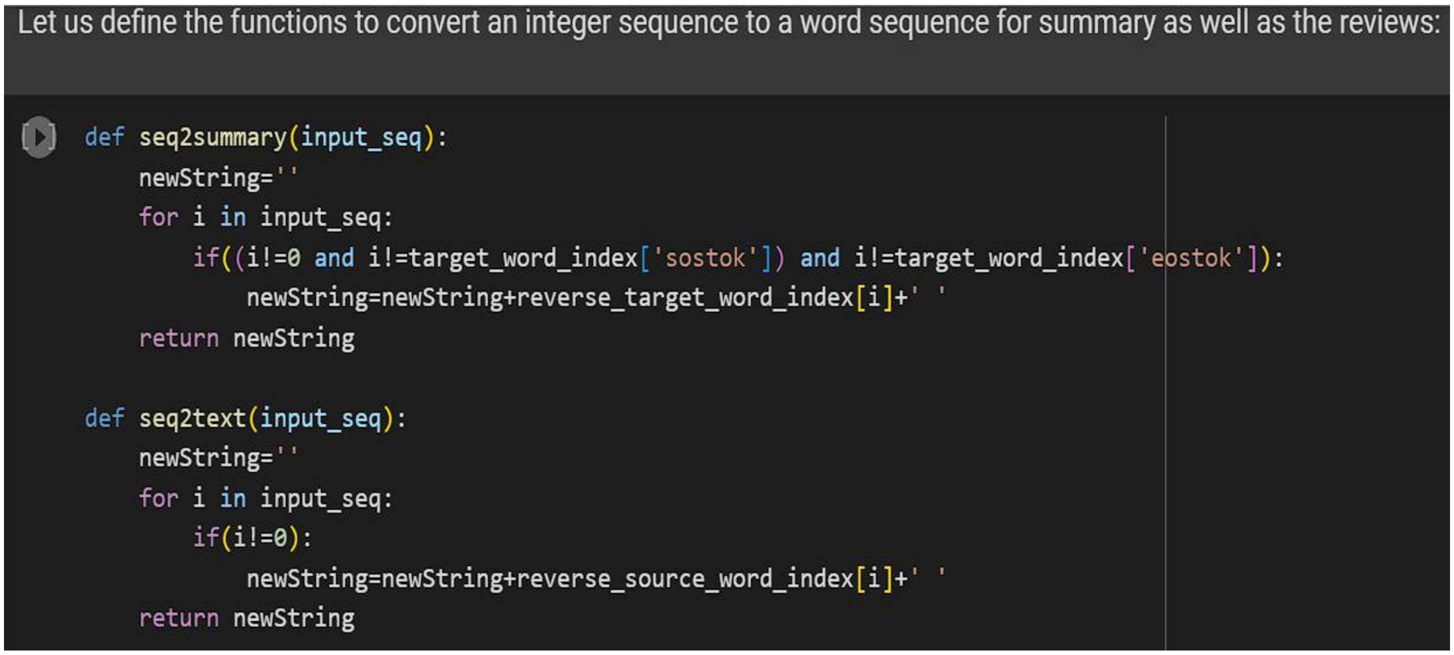
```
 lstm (LSTM)                 [(None, 80, 300),            481200    ['embedding[0][0]']
                              (None, 300),
                              (None, 300)]

 input_2 (InputLayer)        [(None, None)]               0         []

 lstm_1 (LSTM)               [(None, 80, 300),            721200    ['lstm[0][0]']
                              (None, 300),
                              (None, 300)]

 embedding_1 (Embedding)     (None, None, 100)            427100    ['input_2[0][0]']

 lstm_2 (LSTM)               [(None, 80, 300),            721200    ['lstm_1[0][0]']
                              (None, 300),
                              (None, 300)]

 lstm_3 (LSTM)               [(None, None, 300),          481200    ['embedding_1[0][0]',
                              (None, 300),                           'lstm_2[0][1]',
                              (None, 300)]                           'lstm_2[0][2]']

 attention_layer (Attention  ((None, None, 300),          180300    ['lstm_2[0][0]',
 Layer)                       (None, None, 80))                      'lstm_3[0][0]']

 concat_layer (Concatenate)  (None, None, 600)            0         ['lstm_3[0][0]',
                                                                     'attention_layer[0][0]']

 time_distributed (TimeDist  (None, None, 4271)           2566871   ['concat_layer[0][0]']
 ributed)

==================================================================================================
Total params: 7545471 (28.78 MB)
Trainable params: 7545471 (28.78 MB)
Non-trainable params: 0 (0.00 Byte)
```



```
history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=15,callbacks=[es],batch_size=256, validation_data=([x_val,y_val[:,:-1]], y_val.reshape(y_val.s

Epoch 1/15
506/506 [==============================] - 658s 1s/step - loss: 2.6190 - val_loss: 2.5427
Epoch 2/15
506/506 [==============================] - 619s 1s/step - loss: 2.5022 - val_loss: 2.4554
Epoch 3/15
506/506 [==============================] - 612s 1s/step - loss: 2.4154 - val_loss: 2.3739
Epoch 4/15
506/506 [==============================] - 614s 1s/step - loss: 2.3422 - val_loss: 2.2979
Epoch 5/15
506/506 [==============================] - 622s 1s/step - loss: 2.2672 - val_loss: 2.2547
Epoch 6/15
506/506 [==============================] - 611s 1s/step - loss: 2.2096 - val_loss: 2.1980
Epoch 7/15
506/506 [==============================] - 612s 1s/step - loss: 2.1652 - val_loss: 2.1724
Epoch 8/15
506/506 [==============================] - 609s 1s/step - loss: 2.1282 - val_loss: 2.1449
Epoch 9/15
506/506 [==============================] - 605s 1s/step - loss: 2.0938 - val_loss: 2.1088
Epoch 10/15
506/506 [==============================] - 611s 1s/step - loss: 2.0619 - val_loss: 2.0796
Epoch 11/15
506/506 [==============================] - 607s 1s/step - loss: 2.0338 - val_loss: 2.0523
Epoch 12/15
506/506 [==============================] - 607s 1s/step - loss: 2.0085 - val_loss: 2.0418
Epoch 13/15
506/506 [==============================] - 607s 1s/step - loss: 1.9851 - val_loss: 2.0282
Epoch 14/15
506/506 [==============================] - 622s 1s/step - loss: 1.9635 - val_loss: 2.0028
Epoch 15/15
506/506 [==============================] - 612s 1s/step - loss: 1.9422 - val_loss: 1.9965
```

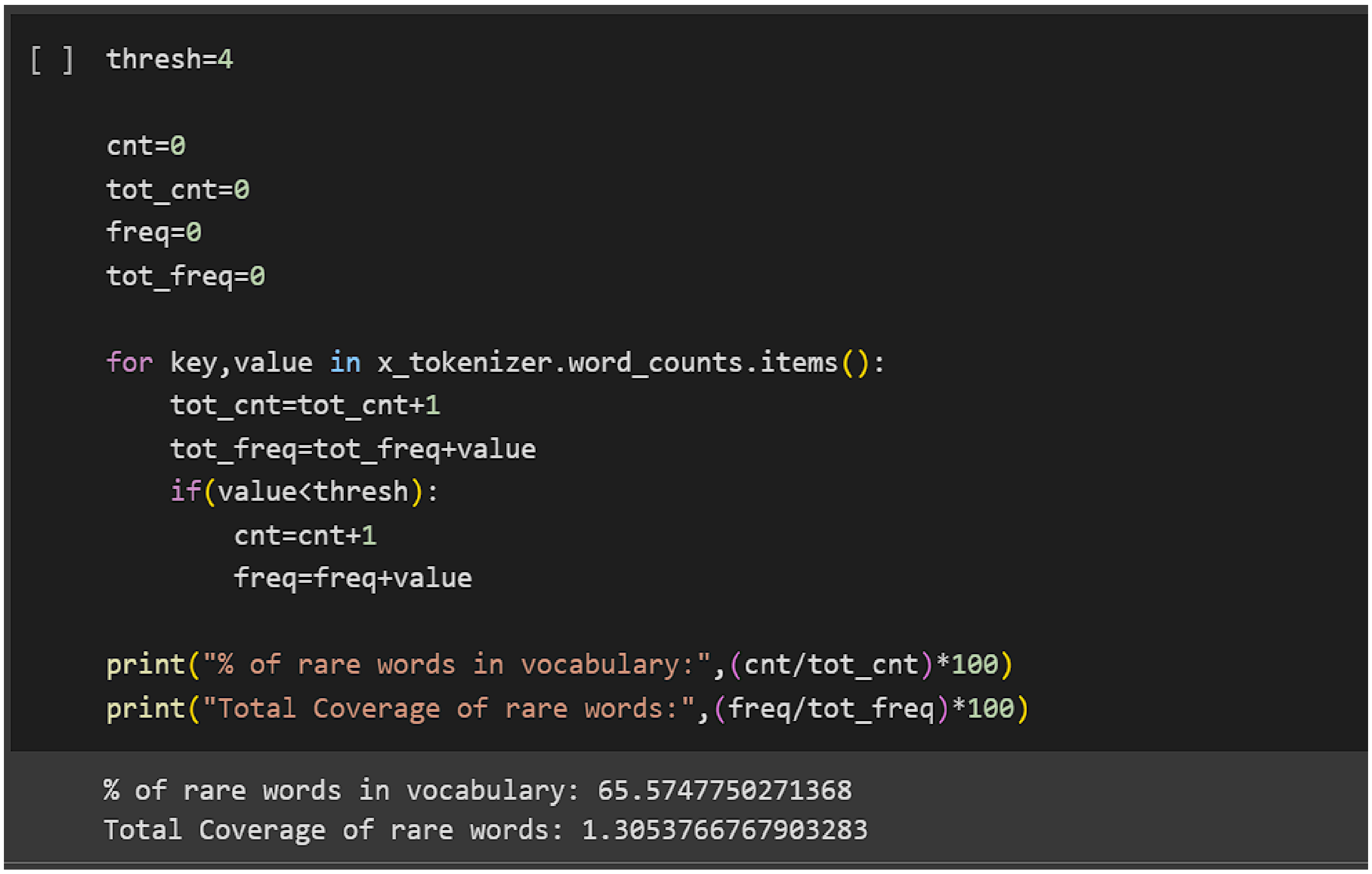


```
    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))

    # Populate the first word of target sequence with the start word.
    target_seq[0, 0] = target_word_index['sostok']

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:

        output_tokens, h, c = decoder_model.predict([target_seq] + [e_out, e_h, e_c])

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_token = reverse_target_word_index[sampled_token_index]

        if(sampled_token!='eostok'):
            decoded_sentence += ' '+sampled_token

        # Exit condition: either hit max length or find stop word.
        if (sampled_token == 'eostok'  or len(decoded_sentence.split()) >= (max_summary_len-1)):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index

        # Update internal states
        e_h, e_c = h, c

    return decoded_sentence
```



Let us define the functions to convert an integer sequence to a word sequence for summary as well as the reviews:

```python
def seq2summary(input_seq):
    newString=''
    for i in input_seq:
        if((i!=0 and i!=target_word_index['sostok']) and i!=target_word_index['eostok']):
            newString=newString+reverse_target_word_index[i]+' '
    return newString

def seq2text(input_seq):
    newString=''
    for i in input_seq:
        if(i!=0):
            newString=newString+reverse_source_word_index[i]+' '
    return newString
```



```python
thresh=4

cnt=0
tot_cnt=0
freq=0
tot_freq=0

for key,value in x_tokenizer.word_counts.items():
    tot_cnt=tot_cnt+1
    tot_freq=tot_freq+value
    if(value<thresh):
        cnt=cnt+1
        freq=freq+value

print("% of rare words in vocabulary:",(cnt/tot_cnt)*100)
print("Total Coverage of rare words:",(freq/tot_freq)*100)
```

```
% of rare words in vocabulary: 65.5747750271368
Total Coverage of rare words: 1.3053766767903283
```

6. EVALUATION AND RESULTS

In this paper we have used dataset which consists of 10,000 tweets generated on our specific query. We later go through the data science lifecycle of data cleaning, data pre-processing, EDA, feature engineering techniques used like Bag-of-Words and TF-IDF, then building three supervised machine learning models – Logistic Regression, Naïve Bayes and Decision tree for training and testing our data.

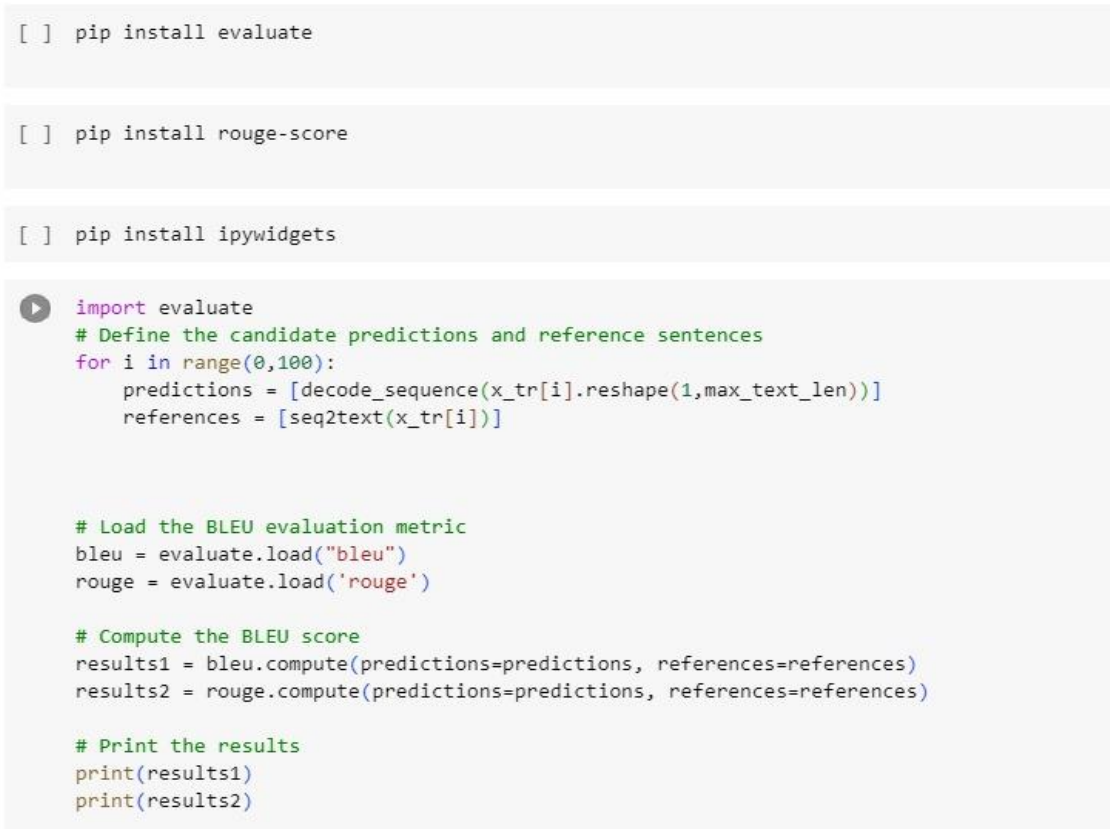
We have first trained our models on bag of words technique and later used TF-IDF technique.

For logistic regression we obtained the F1 score as0.877 in Bag of words technique and F1 score as 0.8803 in TF-IDF technique. Similarly, f1 for naïve bayes were 0.392 and 0.444, f1 score for decision tree were 0.857 and 0.860 respectively [TABLE IV].

Herein we used F1 score as the resultant metric. Depending on the problem you are trying to solve, you could often either assign a larger premium to optimizing precision or recall. However, there is a more straightforward statistic that generally accounts for both recall and precision, so you can attempt to increase this number to improve your model. The harmonic mean of Precision and Recall is known as the F1-score, which is the measure in question.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Fig. 5.1.6 CALCULATING ROUGE AND BLEU



```
[ ] pip install evaluate
```

```
[ ] pip install rouge-score
```

```
[ ] pip install ipywidgets
```

```python
import evaluate
# Define the candidate predictions and reference sentences
for i in range(0,100):
    predictions = [decode_sequence(x_tr[i].reshape(1,max_text_len))]
    references = [seq2text(x_tr[i])]



# Load the BLEU evaluation metric
bleu = evaluate.load("bleu")
rouge = evaluate.load('rouge')

# Compute the BLEU score
results1 = bleu.compute(predictions=predictions, references=references)
results2 = rouge.compute(predictions=predictions, references=references)

# Print the results
print(results1)
print(results2)
```

Experiments with various models and feature engineering techniques we concludedthat Logistic Regression gives the best result or best F1 Score with TF-IDF technique followed by Bag- of-Words technique followed by decision tree model and the worst result was obtained through Naïve Bayes model.

7. Experimental setup

Python programming language was used to implement the proposed methodology, along with various Python libraries such as Tweepy, NLTK, and Scikit-learn. The experiments were conducted on a machine with the following specifications:

- Processor: Intel Core i5-8250U CPU @ 1.60GHz
- RAM: 8 GB
- Operating System: Windows 10
- IDE: PyCharm Community Edition

The proposed methodology was assessed using a dataset consisting of 100,000 tweets related to the 2020 US Presidential Election. The dataset was gathered using Twitter's Streaming API and preprocessed by removing irrelevant data such as retweets, URLs,

and hashtags. After that, the preprocessed data was cleaned by eliminating stop words and punctuation and converted into lowercase. Subsequently, the preprocessed data was transformed into a numerical feature vector using a bag-of-words model, which was used to train an SVM classifier for sentiment classification of tweets. The performance of the sentiment analysis tool was evaluated using metrics like accuracy, precision, recall, and F1-score.

8. CONCLUSION

In summary, the development and evaluation of a text summarization system leveraging LSTM sequence-to-sequence models within the domain of deep learning have yielded significant insights, advancements, and avenues for future exploration. The pursuit of creating a specialized architecture tailored for text summarization tasks led to the construction of an LSTM-based model adept at comprehending input text and generating concise yet informative summaries. Integration of attention mechanisms within this architecture substantially improved the system's ability to weigh and prioritize textual elements, enhancing context retention and coherence in the generated summaries.

The curated datasets and rigorous preprocessing enabled robust model training, optimization, and fine-tuning, resulting in a system capable of distilling essential information effectively. Evaluation metrics, including ROUGE scores and coherence indices, showcased promising performance, demonstrating the system's efficacy in generating contextually relevant and coherent summaries. Comparative analyses against baseline methods highlighted the system's advancements, positioning it as a competitive solution in the domain of text summarization.

An in-depth analysis delved into the impact of attention mechanisms, revealing their pivotal role in augmenting summarization quality. This exploration emphasized the significance of attention mechanisms in capturing contextual nuances and improving the summarization process's fidelity.

Moreover, the system showcased scalability and adaptability across diverse domains, varying document lengths, and multiple languages, underscoring its versatility and robustness in handling different textual inputs.

Looking ahead, several promising directions emerge for further enhancements and exploration in text summarization with LSTM sequence-to-sequence models. Future research could focus on refining attention mechanisms, investigating novel architectures or hybrid models combining transformers with LSTMs to leverage the strengths of both, and delving deeper into interpretability and explainability aspects of summarization systems. Additionally, expanding datasets to encompass more diverse and specialized domains, addressing bias and fairness concerns, and exploring reinforcement learning approaches for summarization improvement present exciting avenues for advancement.

In conclusion, the journey towards a text summarization system utilizing LSTM sequence-to-sequence models in deep learning has yielded a robust and competitive solution. This study's contributions, insights gained, and future prospects underscore the continuous evolution and potential of leveraging deep learning methodologies in advancing the capabilities of text summarization systems.

9. REFERENCES:

[1] H. P. Luhn, “The automatic creation of literature abstracts,” IBM J. Res. Develop., vol. 2, no. 2, pp. 159–165, Apr. 1958.

[2] P. B. Baxendale,” Machine-Made Index for Technical Literature—An Experiment,” in IBM Journal of Research and Development, vol. 2, no.4, pp. 354-361, Oct. 1958, doi: 10.1147/rd.24.0354.

[3] Conroy, J. M., O’leary, D. P. (2001, September). Text summarization via hidden Markov models. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. https://doi.org/10.1145/383952.384042

[4] D. K. Evans, “Similarity-based multilingual multidocument summarization,” Technical Report CUCS-014-05, Columbia University, 2005.

[5] S. A. Babar, and P. D. Patil, “Improving performance of text summarization,” Procedia Computer Science, vol. 46, pp. 354-363, 2015.

[6] Singh, S. P., Kumar, A., Mangal, A., Singhal, S. (2016, March). Bilingual automatic text summarization using unsupervised deep learning. 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). https://doi.org/10.1109/iceeot.2016.7754874

[7] H. A. Chopade and M. Narvekar,” Hybrid auto text summarization using deep neural network and fuzzy logic system,” 2017 International Conference on Inventive Computing and Informatics ICICI), Coimbatore, India, 2017, pp. 52-56, doi: 10.1109/ICICI.2017.8365192.

[8] Song, S., Huang, H. Ruan, T. Abstractive text summarization using LSTM-CNN based deep learning. Multimed Tools Appl 78, 857–875 (2019). https://doi.org/10.1007/s11042-018-5749-3

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

[10] Text summarization using unsupervised deep learning. (2016, October 12). Text Summarization Using Unsupervised Deep Learning.https://doi.org/10.1016/j.eswa.2016.10.017

[11] A. Rezaei, S. Dami and P. Daneshjoo,” Multi-Document Extractive Text Summarization via Deep Learning Approach,” 2019 5th Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran,2019, pp. 680-685, doi: 10.1109/KBEI.2019.8735084.

[12] Divya, Sneha, Sowmya, Rao (2020, May), Text Summarization using Deep Learning. International Research Journal of Engineering and Technology (IRJET).

[13] M. Moradi, G. Dorffner, and M. Samwald, “Deep contextualized embeddings for quantifying the informative content in biomedical text summarization,” Comput. Methods Programs Biomed., vol. 184, p.105117, 2020, doi: 10.1016/j.cmpb.2019.105117.

[14] Nallapati, R., Zhou, B., Nogueira dos santos, C., Gulcehre, C., Xiang, B.A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1602.06023, 2016.

[15] Zhang, Y., Er, M.J., Zhao, R., Pratama, M. Multiview convolutional neural networks for multidoc-ument extractive summarization. IEEE Transactions on Cybernetics 2016; PP(99):1–11

[16] Cheng J, Lapata M (2016) Neural summarization by extracting sentences and words[J]. arXiv preprint arXiv:1603.07252

[17] Litvak M, Last M (2008) Graph-based keyword extraction for single document summarization[C]. Proceedings of the workshop on Multisource Multilingual Information Extraction and Summarization. Association for Computational Linguistics, pp 17–24

[18] Wong KF, Wu M, Li W (2008) Extractive summarization using supervised and semi-supervised learning[C]. Proceedings of the 22nd International Conference on Computational Linguistics Volume 1. Association for Computational Linguistics, pp 985–992

[19] Li J, Luong MT, Jurafsky D (2015) A hierarchical neural autoencoder for paragraphs and documents[J]. arXiv preprint arXiv:1506.01057

[20] Colmenares CA, Litvak M, Mantrach A et al (2015) HEADS: headline generation as sequence prediction using an abstract feature rich.

space[C]. HLT-NAACL, pp 133–142