This is your Turnitin Digital Receipt

Do not reply to this email (via NCI Moodle) <noreply@ncirl.ie>

Thu 12/12/2019 11:56 AM

To: Kaarthic Muthiah <x18129579@student.ncirl.ie>

Dear Kaarthic Muthiah,

You have successfully submitted the file **X18129579_ResearchProj_Report_ConfigManual** to the assignment **Submit here: Research Project Report (DEADLINE: 12/Dec/2019 2pm): Part 1** in the class **MSc Cloud : Research Project** on **12-Dec-2019 11:56AM**. Your submission id is **1233055117**. Your full digital receipt can be viewed and printed from the assignment inbox or from the print/download button in the document viewer.

Thank you for using Turnitin,

The Turnitin Team

Automatic Coherent and Concise Text Summarization using Natural Language Processing

MSc Research Project
Cloud Computing

Kaarthic Muthiah
Student ID:X18129579

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kaarthic Muthiah
Student ID:	X18129579
Programme:	Cloud Computing
Year:	2019
Module:	MSc Research Project
Supervisor:	Divyaa Manimaran Elango
Submission Due Date:	12/12/2019
Project Title:	Automatic Coherent and Concise Text Summarization using Natural Language Processing
Word Count:	7549
Page Count:	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	25th January 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automatic Coherent and Concise Text Summarization using Natural Language Processing

Kaarthic Muthiah
X18129579

Abstract

Over the past two decades, with the advancements in the World Wide Web and Internet, there has been an exponential increase in the amount of online information causing difficulties in retrieving the precise and necessary information quickly. The solution to this problem is the Automatic Text Summarization, one of the important domains of Natural Language Processing (NLP) which is being extensively focused by the research community. Text Summarization helps to shrink the size of the source document and presents only the key features without compromising the overall context of the input document. Summarization is broadly classified into two types - extractive and abstractive summarization depending on how the original content is structured in the final summary. In the past, the researchers concentrated widely on extractive approaches and now there has been a gradual shift in the research trend towards abstractive methods and fusion of both extractive and abstractive ones. Consequently, in this paper, we propose a novel Combined Extractive Abstractive Text Summarization (CEATS) model which integrates the benefits of both extractive and abstractive approaches to achieve more concise, logical and human readable summaries of the online product reviews collected over a period of time. The extractive stage includes word frequency based sentence feature extraction, graph based sentence ranking algorithm whereas the abstractive phase involves deep artificial neural network approach. It consists of sequence to sequence encoder decoder model made of RNN LSTM networks.

Keywords— Natural Language Processing (NLP), Extractive Summarization, Abstractive Summarization, Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM)

Contents

1 Introduction

Natural languages are languages which are commonly used by human beings in their day-to-day life for communicating with each other either in the form of text or speech. For example, English, Spanish. These languages are usually informal in nature and are totally different from the computer programming languages such as C++, Java and Python which have formal syntax (Hingu et al.; 2015). As we all know, most of the data in today's world are in unstructured format which comprises of text, images, audio and video files. In order to get a deep understanding of such textual data, we use the concept of Natural language Processing (NLP). NLP refers to the branch of Artificial Intelligence (AI) which is related to computational linguistics. It helps the computers to learn, read, understand, interpret and obtain meaningful information from human languages. NLP is broadly classified into two components namely - Natural Language Understanding (NLU) and Natural Language Generation (NLG) (Hardeniya et al.; 2016). NLP has a variety of real time applications. A few of them are chatbots, sentimental analysis, spelling check, search engines, information extraction from documents or website, personal voice assistants like Siri, Alexa and Google Assistant, document summarization, auto-complete feature, language translation and spam email or fake reviews classification.

Over the recent years, with the technological advances in the World Wide Web and the Internet, there has been an exponential growth in the amount of online information. Nowadays, people rely the most on the internet for getting information. As they are overloaded with plethora of information from a wide variety of sources, it has become really tough to figure out the relevant information based on the user query. So, there is a serious need for a summarization technique to efficiently access the necessary information, thereby minimizing the reading time and efforts of the users. This helps the users to get a gist of the entire article within a short duration. When a document is being summarized by human beings, they go through the complete article, understand the context and then use the key points to produce their own summary. The quality of such human produced summaries will be exceptional. However, manually summarizing a document consumes more time. (Sethi et al.; 2017).

So, a computer based approach known as Automatic Text Summarization is obviously required for reducing the size of the longer text to produce a compact and coherent summary without compromising the main ideas of the original content using the power of NLP (Saggion and Poibeau; 2013). According to (Radev et al.; 2002), summary is defined as a piece of text that is obtained from one or more texts which presents the predominant information from the original text while retaining the length of the target summary to be either half or less than that. Based on how the contents of the summary are organized, automatic summarization can be categorized into two major types namely, extractive and abstractive text summarization. Extractive text summarization produces the output summary by picking up the most important sentences or phrases from the input text and combining them together. It does not produce any new text and uses only the existing text from the original document. Basically, it is like highlighting the key points in an article. On the other hand, Abstractive text summarization technique is used to generate more concise summaries with novel phrases or sentences by understanding the overall meaning of the input text by using advanced NLP methods. Abstractive summaries are more human like in nature that are paraphrasing in their own words while preserving the main ideas of original article (Saziyabegum and Sajja; 2016).

The various ways of representing the same content has always fascinated the people and it has made the research community to widely focus on the automatic text summarization domain. It has its advent nearly sixty years back in late fifties when they researched about summarizing scientific articles (Saggion and Poibeau; 2013).From then till now, there has always been a lot of improvements in this domain by proposing diverse approaches to the summarization problem. Presently, text summarization is widely used in different areas like online search engines, emails, news articles, research documents, Wikipedia articles, online blogs, product reviews and comments. Human generated summaries are usually abstractive rather than extractive. However, researchers have mostly focused on extractive summarization since it is comparatively easier than abstractive approach. Fully extractive summaries yield better results compared to the abstractive summaries. This is due to the fact that abstractive summarization techniques deals with issues such as semantic representation and natural language generation which are commensurately tougher than data driven methods like sentence extraction (Allahyari et al.; 2017).

1.1 Motivation

In the recent times, the research trend slowly shifted towards abstractive summarization and combination of extractive and abstractive techniques (Gupta and Gupta; 2018).Right now, there is a need for more sophisticated, first-class, coherent and concise summaries that are grammatically correct and knowledge rich. Also, in this modern era of Internet, e-commerce has gained a lot of popularity across different industry sectors such as food and beverages, clothing, travel, electronics, real estate and many more. This has caused people to post and share millions of user reviews, comments and feedback about various products on the e-commerce websites and other online portals or forums. Manually analysing all such reviews to get a better insight about the product will cost the users a lot of time. Another, disadvantage is that it is challenging for the companies to understand the user opinions quickly and improve the quality of those products. This problem has always fascinated me over the recent years.

Subsequently, in this research paper, we propound a two phase approach of integrating both extractive and abstractive summarization methods in order to achieve good quality summaries of the numerous online product reviews. In our proposed model, the output of the extractive phase (first phase) is passed as input to the abstractive phase (second phase) to obtain the final output summary.

1.2 Research Questions

This research paper addresses the following questions as mentioned in our research proposal (Muthiah; 2019):

- How longer texts can be shortened while preserving the vital information and overall essence of the source content to minimize the reading time of the readers?
- How the combined extractive and abstractive summarization model will improve the accuracy of the output summary?

This paper is outlined as follows: Section 2 describes the related work in the field of extractive and abstractive text summarization. Section 3 and Section 4 explains the proposed methodology and approach or design workflow respectively. Section 5 deals with proposed implementation. Section 6 discusses the various experiments performed and evaluation of the output results. And finally, the conclusion and future work.

2 Related Work

This section imparts the literature review of different state-of-the-art existing methods in the realm of automatic text summarization. Basically, the amount of digital content which is present online has been increasing exponentially at a faster pace over the last decade. This has caused hampering of information than what is actually needed for the users. People find it really arduous to identify the necessary information from a variety of sources. Also, it consumes a lot of time to manually go through the unstructured information on the internet and summarize them. Consequently, automatic text summarization has become the need of the hour in the present day. This has always fascinated the research community to focus widely on the summarization problem. In general, text summarization is widely divided into two main types namely, extractive summarization and abstractive summarization. Most of the research works done in the past until early 2000s or the past decade concentrated widely on extractive summarization. But, over the recent years, the paradigm has gradually moved towards abstractive summarization owing to the need of more accurate and less redundant summaries. Discussed below are some of the past works in these two approaches.

As mentioned before, text summarization had its root in late 1950s when (Luhn; 1958) formulated a word or phrase frequency based approach to score the sentences and pick up the top ranking sentences to form the summary of the scientific article. The sentence position feature was used by (Baxendale; 1958) to determine the most appropriate sentences to be used in the summary and also suggested that either the first or last sentence in a paragraph can be used to identify the topic sentence of that paragraph. Ten years later, new features like cue words and title words along with word frequency and sentence position was used to calculate the sentence weights. This method proposed by (Edmundson; 1969) produced a similarity of about 44% between human and machine produced summaries. There are also other word level and sentence level features such as topic sensitive word, biased word, content words (nouns, verbs, adverbs, adjectives), upper case word, paragraph position and sentence length that can be used to extract the sentences for the summary (Chen et al.; 2002). (Vanderwende et al.; 2007) proposed SumBasic system, a word frequency or word probability based technique for selecting the top sentences for the summary and it is based on a greedy approach whereas, (Yih et al.; 2007) and (Alguliev et al.; 2011) employed an optimization approach to increase the appearance of the significant words in the summary.

The tf-idf sentence weighting technique is based on the term frequency and inverse document frequency values and it is proven to be one of the efficient methods to calculate sentence scores. Since calculating these scores are pretty easier and faster, it has been extensively used by a variety of summarizers in the past, out of which a few are listed below. The sentence similarity was measured using the tf-idf scores in the summarization model proposed by (Erkan and Radev; 2004). (Alguliev et al.; 2011) developed a generic unsupervised summarization model which extracts the important content of the source document to a greater extent with less redundancy with the help of tf-isf (term frequency inverse sentence frequency) scores. It can be applied to both single and multi document summarization tasks. Inspired from (Alguliev et al.; 2011), to overcome the optimization issues, a differential evolution algorithm based summarization model was developed by (Alguliev et al.; 2013). A deep ensemble auto encoder unsupervised deep learning model using tf-idf was propunded by (Yousefi-Azar and Hamey; 2017) while (Alami et al.; 2018) also proposed a similar model but with a variational auto encoder focusing on tf-idf values of both local and global vocabularies for arabic document extractive summarization. (Qaiser and Ali; 2018) used the tf-idf scores to determine the significance of the top key words from the various websites in different domains.

Graph based ranking algorithms basically gained popularity from the PageRank algorithm, a web page ranking algorithm from Google (Brin and Page; 1998) which has created a revolution in the internet world. This graph based technique was also incorporated into the domain of document summarization by various researchers as stated below. (Erkan and Radev; 2004) proposed the LexRank algorithm , a graph based ranking technique inspired from the PageRank algorithm. It is used to find the sentence importance based on lexical centrality. TextRank graph based ranking algorithm, an unsupervised method was introduced by (Mihalcea and Tarau; 2004) was used for performing natural language tasks such as keyword and sentence extraction. The same TextRank algorithm was also employed by (Barrera and Verma; 2012) for single document extractive summarization and by (Li et al.; 2019) for keyword extraction for social media short texts. There are other different approaches for extractive summarization which includes Naive Bayes model, log linear model, decision tree approach, latent semantic analysis, cluster based, fuzzy logic based and hybrid approaches such as combining fuzzy based and LSA based approaches (Saziyabegum and Sajja; 2016).

As stated before, abstractive summarization is used to create more shorter, absolute summaries by either paraphrasing or using new words instead of simply extracting the salient portions of the source text. Abstractive summarization is mainly classified into two categories - structure-based and semantic-based approaches. Structure based approach includes tree based, rule based, ontology based, graph based, template based, lead and body phrase based methods. While, semantic based approach consists of multimodal, information-item based, predicate argument based and semantic graph based methods. However, during the recent times, due to the success in the field of machine translation, a variety of deep learning models and techniques have also been incorporated in the abstractive summarization domain (Gupta and Gupta; 2018). This is because deep learning is capable of retrieving both semantic and structural information from the text. A few of the recent abstractive summarization research works are discussed below.

A Recurrent Attentive Summarizer (RAS) model, which is based on a convolutional attention based conditional encoder and standard RNN decoder architecture that concentrates on each step of the output generation was introduced by (Chopra et al.; 2016). Following a similar method, (Nallapati et al.; 2016) proposed a complete RNN sequence to sequence model with GRU-RNN based networks for both encoder and decoder. This model was used to resolve the keyword capturing, modeling and hierarchical sentence to word structure issues. (See et al.; 2017) presented a new model using pointer generator and coverage mechanism along with the standard sequence to sequence attentional encoder decoder model. Pointer generator technique is used to effortlessly select the important content from the original text and reproduce it with accurate information in the output summary. Also, coverage concept is used to minimize the repetition of words or phrases in the summary. A Selective Encoding Abstractive Sentence Summarization model (SEASS) with an encoder, selective gate network and a decoder which

Extractive Summarization Paper/Author	**Technique Used**
Luhn	word or phrase frequency approach
Vanderwende et al	word frequency based on greedy approach
Yih et al	word frequency with optimization approach
Alguliev et al	Term Frequency-Inverse Sentence Frequency (TF-ISF)
Erkan and Radev	Term Frequency-Inverse Document Frequency(TF-IDF)
Yousefi Azar and Hamey	Deep ensemble auto encoder with TF-IDF
Alami et al	Variational auto encoder with TF-IDF
Qaiser and Ali	Term Frequency-Inverse Document Frequency(TF-IDF)
Brin and Page	Graph based PageRank Algorithm
Erkan and Radev	Graph based LexRank Algorithm
Abstractive Summarization Paper/Author	**Technique Used**
Nallapati et al	Complete RNN seq2seq encoder decoder model with GRU-RNN
See et al	Standard seq2seq encoder decoder with pointer generator mechanism
Zhou et al	Selective encoding encoder decoder model
Hou et al	Standard encoder decoder sequence model with joint attention and sub word method
Fan et al	Convolutional Neural Network sequence to sequence encoder decoder model
Our Research Technique Used	TF-IDF + Graph based Page/Text Rank + Seq2Seq Encoder Decoder Model

Table 1: Summary of Literature Review

is an extension of basic encoder decoder sequence to sequence model was formulated by (Zhou et al.; 2017). The selective gate network controls the flow of encoded information to the decoder thereby minimizing the efforts of decoder in generating the final output summary. (Hou et al.; 2017) used the standard encoder decoder sequence model with a joint attention mechanism for single document summarization which focused on the output sequence as well, thus reducing the word repetition problem. Also, it used a sub word method which helped to handle the rare and unknown words. (Fan et al.; 2017) proposed a convolutional neural network based encoder decoder sequence to sequence model for controllable single document abstractive summarization that allows the user to set their preferences such as style, length and interested entities.

As aforesaid, only in the very recent times, the research focus has turned towards abstractive summarization and combining extractive and abstractive approaches (Gupta and Gupta; 2018) which includes the following. (Hsu et al.; 2018) proposed a combined model which uses both sentence level and word level attentions. It also introduced a new inconsistency loss function which is used to improve the consistency between the two attentions. (Subramanian et al.; 2019) used two neural models namely hierarchical seq2seq pointer and sentence classifier in the extractive model and a Transformer Language model for abstractive phase conditioning on both extracted sentences and whole or portion of the source document. These models were evaluated on large scale lengthy document datasets such as CNN, Daily Mail, Newsroom, PubMed and BigPatent corpus.

On reviewing the past literature in the text summarization domain, it is clear that there has been a large number of researches performed on extractive and abstractive summarization while a very limited number of works in integrating both approaches. Consequently, in this paper as already mentioned in our research proposal (Muthiah; 2019) and in the motivation section above, we implement a combined extractive and abstractive model for summarization and also attempt to evaluate it with a new dataset consisting of online food reviews from Amazon. We use tf-idf and graph based ranking algorithm in the extractive phase and sequence to sequence encoder decoder RNN network in the abstractive model to achieve more accurate, concise and human readable summaries of online reviews.

3 Methodology

This section describes the methodology adopted to approach the implementation of our proposed solution. In this research, since we deal with text summarization, a machine learning problem which is one of the applications of Natural Language Processing (NLP), building a proper and efficient model is a significant task. This can be achieved by following different data analytics frameworks which are a set of well defined processes for successful product delivery. Few popular data science methodologies (Azevedo and Santos; 2008) include Cross Industry Standard Process for Data Mining (CRISP-DM), Knowledge Discovery in Databases (KDD) and Sample, Explore, Modify, Model, Assess (SEMMA).

After a detailed study of the above mentioned methodologies, the one that best fits our research purpose is KDD because it has almost all the steps closely matching with our proposal. KDD[1] focuses mainly on task execution right from data selection or preparation to model building and evaluation rather than the project management procedures and helps in extracting useful information or knowledge from the internet or huge databases or repositories. The KDD methodology comprises of 5 major stages namely: (i)Data Selection (ii)Data Pre-processing (iii)Data Transformation (iv)Data mining and (v) Evaluation or interpretation of results (Fayyad et al.; 1996). Figure 1 illustrates how our proposed approach is aligned with each stage of the KDD methodology.

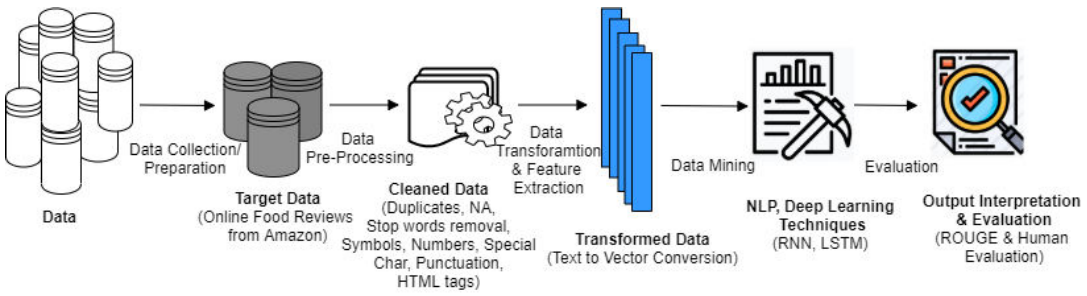

Data
Data Collection/ Preparation
Target Data (Online Food Reviews from Amazon)
Data Pre-Processing
Cleaned Data (Duplicates, NA, Stop words removal, Symbols, Numbers, Special Char, Punctuation, HTML tags)
Data Transforamtion & Feature Extraction
Transformed Data (Text to Vector Conversion)
Data Mining
NLP, Deep Learning Techniques (RNN, LSTM)
Evaluation
Output Interpretation & Evaluation (ROUGE & Human Evaluation)

Figure 1: Proposed methodology in alignment with KDD framework

3.1 Data Collection or Preparation

In this research, the data is prepared from the online fine food reviews from Amazon (McAuley and Leskovec; 2013). It consists of various details such as username, user ID, product information, ratings, product reviews text and a reference summary for those user reviews. All the above information are extracted and transferred into a csv file for further processing. The entire dataset consists of approximately 570,000 reviews and we can select the data samples, say 50,000 or 100,000 records for our implementation purpose based on the computational power or resource available.

[1] http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html

3.2 Pre-processing

Once the input data for our implementation is collected, the next very essential step before starting the data mining process is the data pre-processing. Pre-processing helps in eliminating the noise in the data, that is, irrelevant or unnecessary data from the samples. Such noisy data may have serious impact on model performance and the output results. First, the necessary attributes from the data samples are taken. In this case, we take food reviews text and its reference summary. Then, some data pre-processing steps such as removing duplicates and NA values, removing special characters, symbols, numbers and punctuation, removing stop words and short words, removing HTML tags, contraction mapping and converting everything to lowercase are performed on the selected fields from the sample. All the above data cleaning tasks are performed using Python programming language and its related libraries will be discussed in detail in the upcoming sections of this report.

3.3 Feature Extraction and Transformation

The next step after data cleaning is feature extraction and transforming the input text into numerical and vectorize it, on which various NLP and deep learning techniques or algorithms can be applied. Here we use frequency of words feature to calculate the sentence scores with help of Term Frequency-Inverse Document Frequency (TF-IDF) method. This feature extraction and data transformation is achieved using the scikit-learn[2] (sklearn) python machine learning library.

3.4 Data Mining

In this step, we use different data mining techniques to build different models to achieve our objective. The aim of this research is to build a combined extractive and abstractive text summarization model using various NLP and deep learning techniques. We use TextRank algorithm, which is a graph-based ranking algorithm in NLP that is basically inspired from Google's PageRank algorithm. And we use skearn, tensorflow[3] and keras[4] machine learning and deep neural network python libraries to develop models using RNN and LSTM.

3.5 Evaluation

After the building the model, it needs to be trained and validated. The input textual data is divided in 80:20 ratio for training and testing the model respectively. We perform model training by tuning the hyper-parameters such as number of epochs, optimizer, batch size, loss function, embedding dimension and learning rate. In each case, the training and testing loss and accuracy are determined and plotted as a graph to get a better understanding of the condition under which model is trained properly and produce more accuracy and less loss comparatively. And, the final predicted output summary for the input food reviews text is compared with the given reference summary and evaluated using ROUGE metrics (Recall Oriented Understudy for Gisting Evaluation). It is a benchmark metric for obtaining the quality of the model generated summary.

[2] `https://scikit-learn.org/stable/`
[3] `https://www.tensorflow.org/`
[4] `https://keras.io/`

4 Design Specification

This section discusses about the design and workflow of our CEATS model. It also describes the various components used to build the model.

4.1 CEATS Model

In this paper, we present an automatic text summarization model known as Combined Extractive and Abstractive Text Summarization (CEATS) model comprising of two major phases namely: (i) Extractive phase and (ii) Abstractive phase. The output of the first phase is used as the input to the second phase. The below Figure 2 demonstrates the workflow of our summarization model.

4.2 Stage 1: Extractive Approach

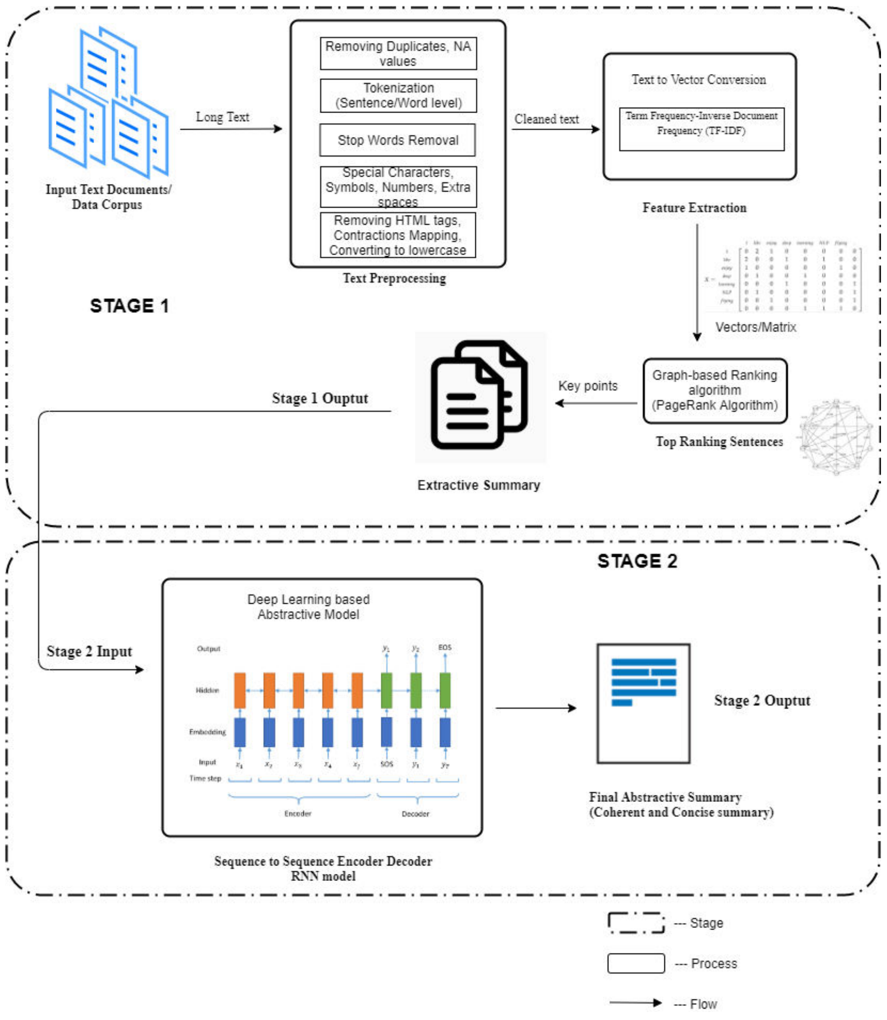
The very basic step for our summarization process is input data collection. in our case, the dataset is prepared from the online fine food reviews from Amazon. Once the data corpus is ready, it is fed into the summarization model to predict the summary of the input text. The 3 key steps in this phase are the following:

(i) Text Pre-processing:

Raw data or real-world data will have lot of unnecessary, meaningless, inconsistent and unstructured data which will be really arduous to be interpreted and processed by the machines. This is referred as noise in the data. Thus, data cleaning or pre-processing is mandatory for any data mining tasks in order to get proper results. Firstly, remove the duplicates and NA values from the input food reviews. Now, the remaining reviews are splitted into a list of sentences and this process is known as sentence tokenization. Next from those sentences, we will remove the stop words or more common words, special characters, symbols, numbers, extra spaces, HTML content and contractions mapping. Finally, convert all the text to lowercase.

(ii) Feature Extraction

In this step, the cleaned text is used to extract useful features from it and transform them into vectors to be used by the NLP algorithms to find the important sentences to be presented in the summary. In our case, we use the word frequency feature with the help of Term Frequency-Inverse Document Frequency (TF-IDF) (Allahyari et al.; 2017) technique which is widely used for summarization and information retrieval problems. It is employed to determine the sentence scores on the basis of the TF-IDF score of words in those sentences. Basically, Term Frequency (TF) is defined as the number of times a particular word appears in a text document. If suppose, there are many documents in the input data corpus each with different lengths, the TF is divided by the total number of words in the document. Inverse Document Frequency (IDF) of a particular word is given by total number of documents divided by number of documents having that word.



Input Text Documents/ Data Corpus
Long Text
Removing Duplicates, NA values
Tokenization (Sentence/Word level)
Stop Words Removal
Special Characters, Symbols, Numbers, Extra spaces
Removing HTML tags, Contractions Mapping, Converting to lowercase
Text Preprocessing
Cleaned text
Text to Vector Conversion
Term Frequency-Inverse Document Frequency (TF-IDF)
Feature Extraction
Vectors/Matrix
STAGE 1
Graph-based Ranking algorithm (PageRank Algorithm)
Top Ranking Sentences
Key points
Stage 1 Ouptut
Extractive Summary
STAGE 2
Stage 2 Input
Deep Learning based Abstractive Model
Output
Hidden
Embedding
Input
Time step
SOS
EOS
Encoder
Decoder
Sequence to Sequence Encoder Decoder RNN model
Stage 2 Ouptut
Final Abstractive Summary (Coherent and Concise summary)
--- Stage
--- Process
--- Flow

Figure 2: CEATS Model Workflow

The Term Frequency (TF) is given by,

$$TF_{(w,D)} = \frac{(Number\ of\ times\ a\ word\ `w'\ occurs\ in\ a\ document\ D)}{(Total\ number\ of\ words\ in\ that\ document)} \tag{1}$$

Inverse Document Frequency (IDF) is given by the formula,

$$IDF = log(X/D(w)) \tag{2}$$

where,
X – total number of documents in the input data corpus
D(w) – number of documents containing the word ‘w’

The TF-IDF score is given by,

$$TF - IDF_{(w,D)} = eq(1) * eq(2)$$

Therefore,

$$TF - IDF_{(w,D)} = TF_{(w,D)} * log(X/D(w)) \tag{3}$$

(iii) Top Ranking Sentences and Extractive Summary:

In this step, the vectorized format of the sentences are translated into a graphical representation by using TextRank algorithm which is derived from the PageRank algorithm by Google (Erkan and Radev; 2004). This is a graph-based ranking algorithm. The nodes of the graph denote the sentences whereas the edges connecting the nodes represent the similarity scores between the sentences.Then, based on certain threshold value, the top ranking sentences are picked up and concatenated to form the output summary which is extractive in nature.

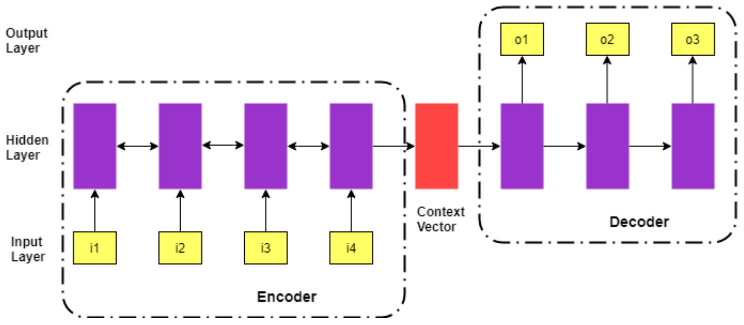


Figure 3: Sequence-to-Sequence Encoder-Decoder Model

4.3 Stage 2: Abstractive Approach

The extractive summary obtained from the stage 1 of our model is used as input for stage 2 - the abstractive phase. We employ a deep learning technique using neural networks in this phase. Few common types of neural networks include Convolutional Neural Network(CNN) and Recurrent Neural Network(RNN). Generally, CNNs are preferred in applications such as image classification, and facial recognition where the input size is fixed (ex: an image) and output of each state depends only on the current state input. On the contrary, ours is text summarization which is a sequential learning problem in which both input and output of the model are a sequence of texts. So, Sequence-to-Sequence Encoder-Decoder model shown in Figure 3 are the best suited for this purpose. We use RNN for both encoder and decoder networks because they use the information retained from all the previous state outputs in its internal memory and current input in order to predict the current state output. However, RNNs are only effective for very short sequences whereas Long Short Term Memory (LSTM), an improvised version of RNN helps to overcome the long term dependency issues (Gupta and Gupta; 2018). Also, we utilize Attention mechanism that helps to focus only on certain portions of the input sequence to predict the output in case of long sequences (Bahdanau et al.; 2014). Hence, the stage 2 of our summarization model comprises of a Sequence to Sequence Attentional Encoder Decoder LSTM RNN. Here, the encoder reads the entire input sequence and encodes into a one hot fixed length context vector which will then be decoded by the decoder network to generate the final output summary.

5 Implementation

In this section, we have discussed about the environmental setup or configuration , different tools, software and libraries used for implementing our proposed model. Also, a detailed description about the dataset used is also presented here.

5.1 Environmental Setup or System Configuration

The implementation of our CEATS summarization model was done using Python (version 3.6.9) programming language. We preferred Python because it is very easy to use, most widely used for machine learning and NLP projects, has wide range of libraries that can be readily imported and also has a good support from the online community and forums. We experimented a lot on different IDEs (PyCharm, Anaconda Jupyter, Spyder) on the local machine *i5 6th GEN, 64-bit Windows OS, 8GB RAM.* Owing to the poor performance and system hangup issues, we decided to use Google Colab. To make the executions faster we use Google Colaboratory popularly known as *"Google Colab"*, a free Jupyter notebook environment that runs completely on Google Cloud and uses Google Compute Engine backend for all the computations. It is very easy to write and execute the code as it does not require any installation on your local machine. All you need is a web browser to access the colaboratory. Colab offers both GPU and TPU hardware accelarators for free with our Google free tier account up to a certain computational power. That is, we can change the runtime type while executing our code. The free GPU offered is *1xTesla K80, 2496 CUDA cores* and free TPU includes *TPU v2, 8 cores, approx 12 GB RAM with a maximum RAM limit up to 36 GB.* Since, TPU is comparatively faster than GPU, we used the TPU accelarator.

5.2 Dataset Description

The data used for this research is extracted from the online fine food reviews from Amazon (McAuley and Leskovec; 2013). It is a public domain dataset under CC0 license and consists of approximately 500,000 reviews covering a span of more than 10 years starting from 1999 till 2012 with details such as user and product information, reviews and their ratings. It includes reviews of nearly 75,000 products posted by over 250,000 users. These interesting statistics[5] about our dataset are summarised below in Table 3. The dataset consists of about 10 attributes as described in the below table Table 2.

Attributes	**Description**
Id	Row Serial No.
ProductId	Unique id for the products
UserId	Unique id of the user
ProfileName	User Name
Helpfulness	Ratio of users who were benefitted from the reviews
Score	Product ratings on a scale of 1 to 5
Time	Review Timestamp
Summary	Summary of the user review
Text	Product Review/Comment

Table 2: Dataset Attributes Description

5.3 CEATS Model Implementation

First of all, the online reviews data is extracted from SNAP[6] (Standford large Network Dataset Collection). Then, from the extracted food reviews text file, we parse the various attributes

[5] `https://snap.stanford.edu/data/web-FineFoods.html`
[6] `https://snap.stanford.edu/data/`

Dataset Statistics	
No. of reviews	568,454
No. of users	256,059
No. of products	74,258
Users (greater than 50 reviews)	260
Median no. of words per review	56
Time Interval	October 1999 - October 2012

Table 3: Dataset Statistics

mentioned in Table 2 and transform it into a CSV file with help of *pandas*[7] python library. From the above mentioned dataset attributes, we will use only *'Text'* and *'Summary'* fields for performing the task of text summarization. The dataset CSV file was imported into DataFrames with help of pandas library and the two necessary attributes were extracted and used for further manipulation. We considered a sample of 100,000 reviews for text pre-processing. We first removed the duplicates and NA values from the sample resulting in nearly 88,000 reiews. Then, we use *sent_tokenize* module from the *nltk.tokenize* package of the NLTK[8] python library. Following this, we remove the stop words, HTML content, remove the special characters, symbols, numbers, extra white-spaces using *NLTK, BeautifulSoup and re(RegEx)*[9] libraries. Also, all the text is converted to lowercase using lower() string handling function within python.

Once data cleaning is completed, the next step is to extract features from the data. Here we use tf-idf term weighting scheme to transform the text input into vector representations. This is accomplished by importing the *feature extraction*[10] module from the *Scikit-learn*(sklearn) python machine learning library. The modules *CountVectorizer* and *TfidfTransformer* within the feature extraction package is used for converting the text into count matrix and then transform it into a normalized tf-idf vector format. Then, the vector representation is converted into a graph with the use of pagerank algorithm from the *NetworkX*[11] python library for creating and studying graphs and networks. Following this, the top ranking sentences are extracted based on the similarity scores between the sentences and presented in the extractive summary as the output of phase 1. This output is loaded into a dataframe along with the reviews source text, reference summary which in turn is converted into a CSV file which will be used for the next phase.

In the second phase, abstractive summarization neural network model is built using *TensorFlow*[12] and *Keras*[13] machine learning and neural networks python libraries. Here, we consider about 50,000 samples from nearly 88,000 reviews resulted from stage 1. First, we need to perform similar pre-processing tasks on the extractive summary output and reference summary as we did in the first phase in order to be used by our machine learning model. Fix the maximum cleaned text and summary lengths based on the distribution of sequence lengths from the chosen sample. Add *'sostok'* START and *'eostok'* END tokens to the reference summary as this will help the model to determine when the sequence starts and ends respectively. The dataset is split into 80% training data and 20% testing data. Now, both the training and testing data are tokenized to form the vocabulary and converted the word sequences into equal length integer sequences by using *Tokenizer* and *pad_sequences* modules from keras.preprocessing package. The

[7] https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

[8] https://www.nltk.org/

[9] https://docs.python.org/3/library/re.html

[10] https://scikit-learn.org/stable/modules/feature_extraction.html#feature-extraction

[11] https://networkx.github.io/documentation/networkx-1.10/index.html

[12] https://www.tensorflow.org/

[13] https://keras.io/

model built here is a three layered LSTM encoder network and a single layered LSTM decoder network with an embedding layer on both encoder and decoder networks , custom attention layer to remember the lengthy sequences and the output layer uses SoftMax activation function. The hidden layers have a dimension of 300 units and embedding layers have a size of 200. Also, a dropout value of 0.4 is used in each hidden layer in order to reduce model overfitting and improve its performance. These layers are implemented and model is built by using various wrappers such as Input, LSTM, Embedding, Dense from the *keras.layers* and *keras.models* packages. One of the hyper-parameters, loss function used here is sparse_categorical_crossentropy because we use integer targets and they will be converted into a one hot vector easily without any memory issues by using this loss function.

After building the model, it needs to be compiled and trained which is achieved by *Model.compile* and *Model.fit* keras models functional methods. Model training is performed by tuning various hyper-parameters such as epochs, optimizer, batch size, embedding dimension, activation, loss function and learning rate and variations in the accuracy and loss values are determined and analyzed. After training phase comes the inference phase, in which we input the testing data to our model and get the output predicted summary. Various experiments performed during model training by hyper-parameter tuning and evaluation of predicted output by our model will be discussed in detail in the upcoming sections of the report.

6 Experiments and Evaluation of the Result

This section discusses about various experiments performed during model training and their results. Also, the different evaluation metrics used for evaluating the model predicted output summary is also presented here.

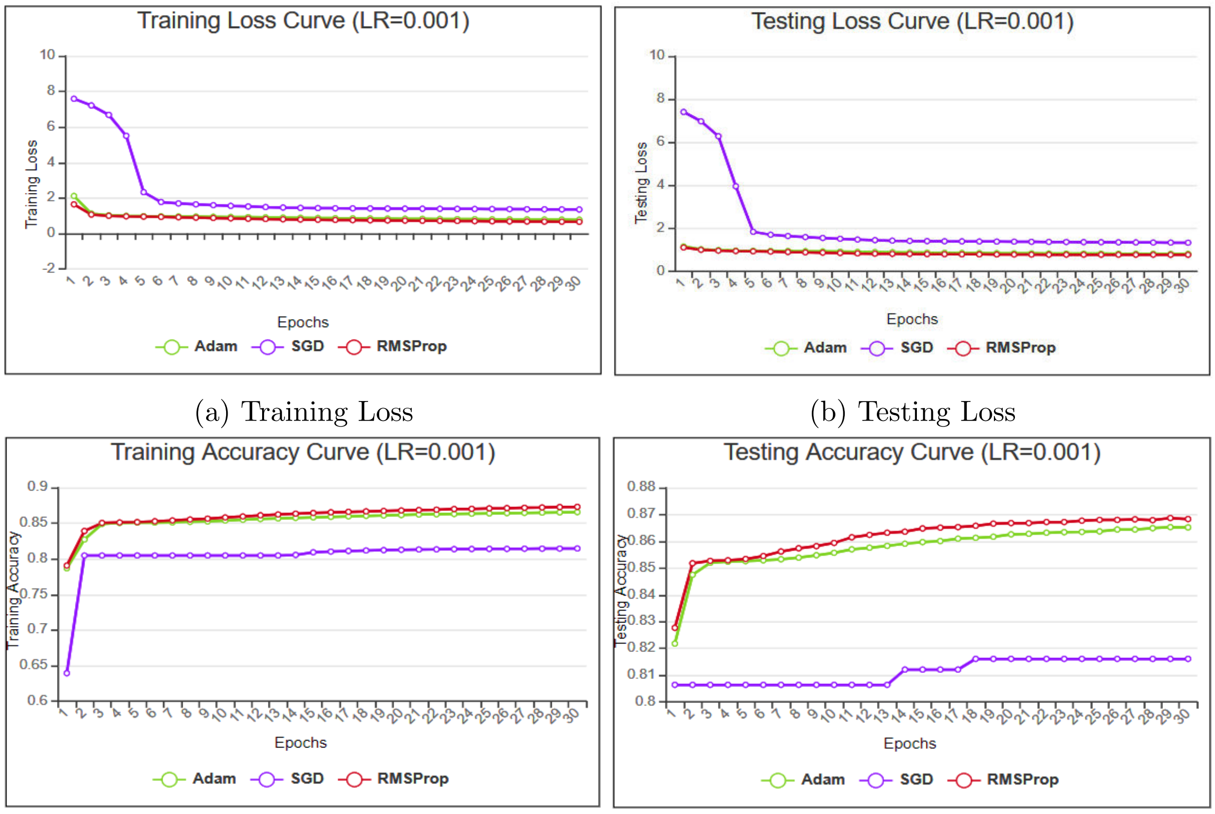
6.1 Model Training Experiments

After cleaning the 50,000 input samples for phase 2, we have about 48,227 samples in the dataset. As mentioned earlier, they are divided into 80:20 ratio for training dataset (38,606 samples) and testing or validation dataset (9621 samples) respectively. Now, the model is trained on the training dataset and is validated using the testing dataset. We carried out different experiments by varying the hyper-parameters to determine the comparatively better performing model with better accuracy. Learning Curves are mathematical notations that are extensively used in the field of machine learning, especially to interpret the performance of the deep learning neural network models over a period of time and for understanding the learning process (Anzanello and Fogliatto; 2011). We plot the loss and accuracy curves to understand the model behaviour over multiple epochs or iterations in each of the experiments performed below. Generally, loss should be minimal and accuracy should be higher.

6.1.1 Experiment 1

In this experiment, we considered the following hyper-parameters: batch size = 1024, epochs = 30, embedding dimension = 200, activation = SoftMax, loss = sparse categorical crossentropy and hidden layer units = 300. For the above values, learning rate = 0.001 is fixed and optimizer values are varied to Adam, SGD, RMSProp and the corresponding training and testing loss and accuracy curves are plotted for the three cases as shown in Figure 4 and are analysed. The graph is plotted with number of epochs or iterations on the x-axis and either loss or accuracy values on the y-axis. figure 4(a), it is clear that the model learns comparatively well with RMSProp optimizer with a low training loss at a rate of 0.001 than the other optimizers, Adam and SGD. In figure 4(b), the model with RMSprop optimizer has most generalized behaviour on the validation dataset after learning the patterns in the training data than the other two cases.

In figure 4(c) and 4(d), both training and validation accuracy has the same increasing trend for RMSprop and outperforms both Adam and SGD optimizers with an accuracy of 86.83% on the testing dataset.



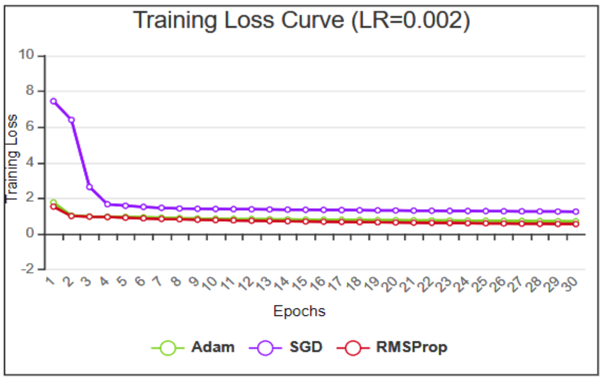
(a) Training Loss (b) Testing Loss

(c) Training Accuracy (d) Testing Accuracy

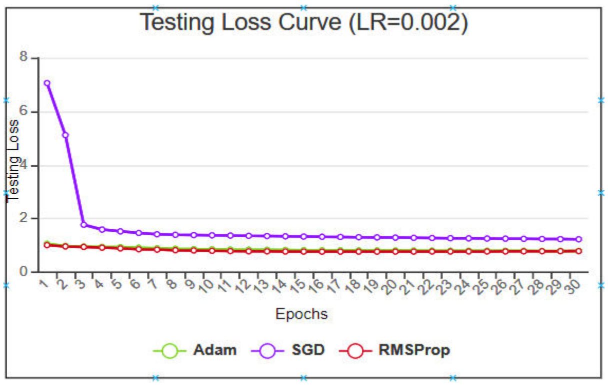
Figure 4: Exp 1 - Model Performance Metrics with Learning Rate 0.001

6.1.2 Experiment 2

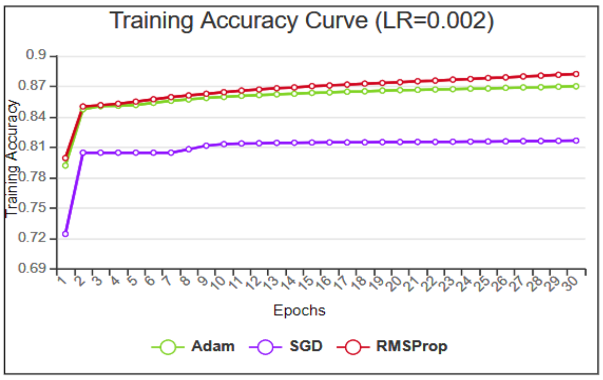
In this case, the following hyper-parameters were taken into account: batch size = 1024, epochs = 30, embedding dimension = 200, activation = SoftMax, loss = sparse categorical crossentropy, hidden layer units = 300. The learning rate is maintained at 0.002 and optimizer values are adjusted to Adam, SGD, RMSProp and respective loss and accuracy values on both training and validation datasets are presented in Figure 5. Even this experiment shows a similar behaviour as the first one. The model outshines with the RMSprop optimizer which has slightly higher performance than Adam optimizer. The model performs the least with SGD optimizer as illustrated in figures 5(a) and 5(b). The validation accuracy of the model with RMSprop is about 86.83% which is just 0.07% higher than that of Adam optimizer as shown in figure 5(d). However, the overall model performance with RMSprop optimizer beats the other two in this experiment.



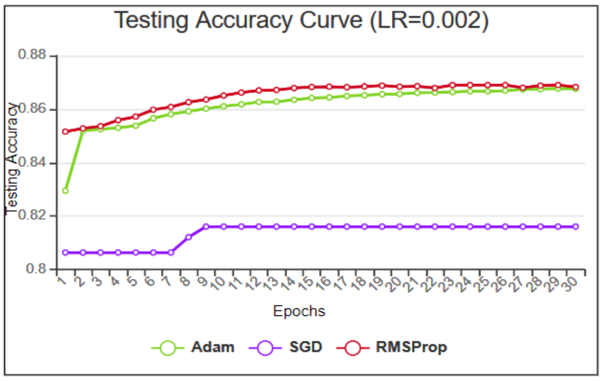
(a) Training Loss



(b) Testing Loss



(c) Training Accuracy



(d) Testing Accuracy

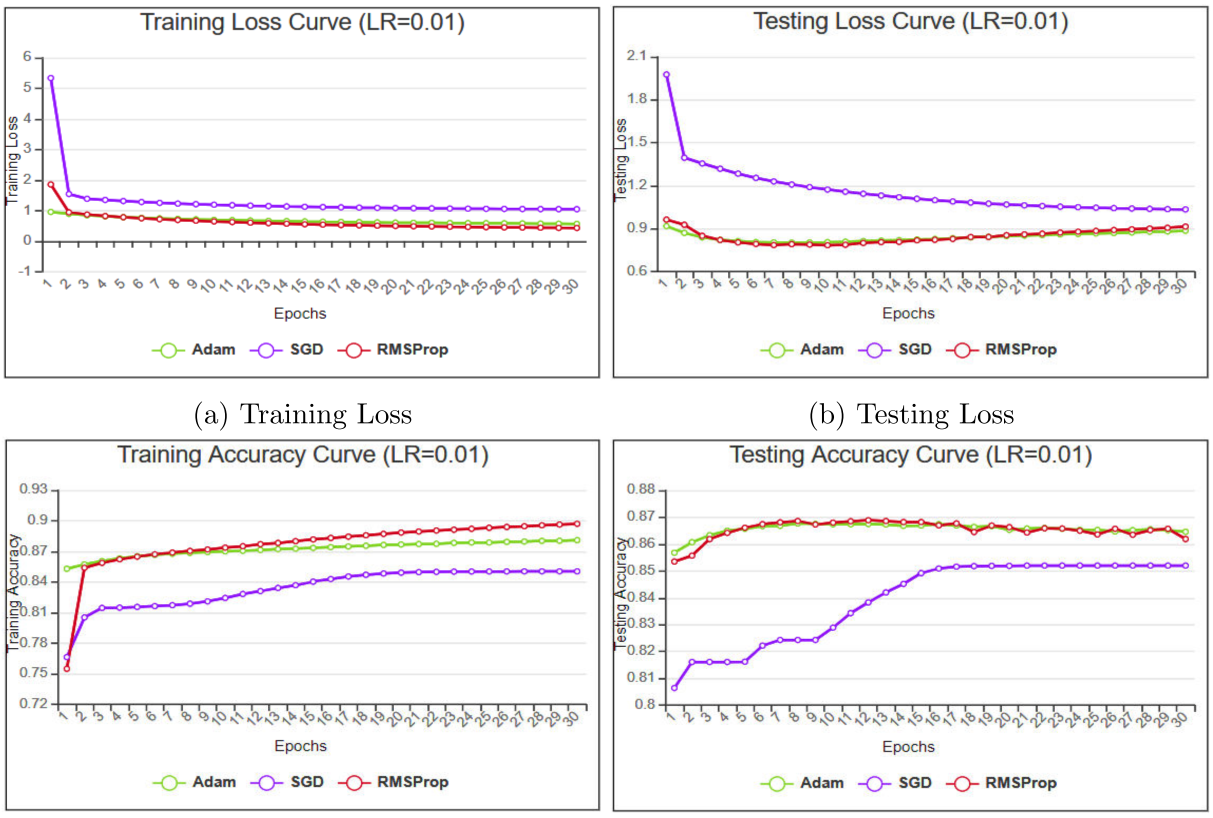
Figure 5: Exp 2 - Model Performance Metrics with Learning Rate 0.002

6.1.3 Experiment 3

As followed for the above two experiments, we use same hyper-parameter values except for the learning rate which is tuned to a value of 0.01. Then, by varying the optimizer function to three different values as done in previous experiments, the loss and accuracy curves are generated as illustrated in Figure 6. Here, from figure 6(a), it can be seen that the model learns well on the training datset with RMSprop optimizer compared to the other two. However, on the contrary, the model performance is good with better generalization behaviour on the validation dataset in case of Adam optimizer with a low testing loss as depicted in figure 6(b). From figure 6(d), it is clear that the model has a better validation accuracy of 86.46% using Adam optimizer when compared to the model using RMSprop (86.19%) and SGD (85.2%) on the testing dataset. Thus, in this case, the model with Adam optimizer surpasses that of SGD and RMSprop.

6.1.4 Discussion

In this research, Natural Language Processing and Deep Learning techniques are used to summarize the online food review comments and provide a gist of it which reduces the reading time of the user and at the same time helps in understanding the overall context of the review easily. In order to obtain a proper fit model, picking up the right hyper-parameters is an important criteria. So, we performed various experiments by tweaking the hyper-parameters, mainly optimizer function and learning rate as discussed above. We could see that the model performed well using RMSProp optimizer in two (Exp 1 and Exp 2) of the above three experiments and had a low loss and high accuracy values. Now, let us examine the model behaviour from the learning curves shown in Figure 7 and analyze its fitness in those two top performing cases. In general,

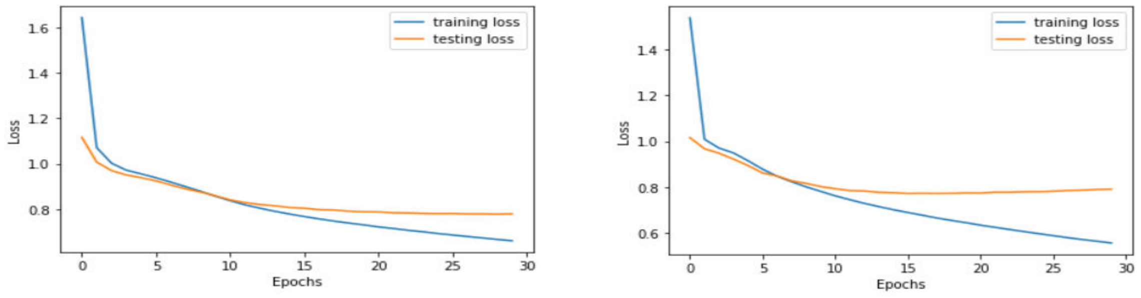


(a) Training Loss (b) Testing Loss

(c) Training Accuracy (d) Testing Accuracy

Figure 6: Exp 3 - Model Performance Metrics with Learning Rate 0.01

we can observe 3 common aspects from the learning curves namely, overfitting, underfitting and good fit (Ian and Yoshua; 2016).



(a) RMSProp with LR = 0.001 (b) RMSProp with LR = 0.002

Figure 7: Learning Curves - Model Behaviour with RMSProp at LR = 0.001 and 0.002

Underfitting is a condition in which the model is not able to effectively learn the training data and can be perceived from the training loss curve. Overfitting refers to the situation in which the model learns the training data very well along with the random variations and noise in the data. Thus, it reduces the ability of the model to generalize on the new data and this condition can be observed from the training loss which keeps on decreasing and validation loss that decreases up to a certain point and then starts increasing. A good fit is a state which is intermediate between underfitting and overfitting. This condition is detected when both training and testing loss decreases until a stable point and has a very minimum gap between the final values of training and validation losses (James et al.; 2013). Comparing the loss learning curves in figures 7, the one shown in 7(a) closely matches with the good fit condition and 7(b) is more aligned towards overfit state. This is because the training and validation losses decreases to a particular stability point and the validation loss has only a very small gap from the training loss until the final epoch as plotted in 7(a). Whereas, in 7(b), training loss keeps on decreasing with number of epochs and validation loss falls upto a particular point after which it rises again steadily.Thus, in conclusion, the model using RMSProp optimizer at a learning rate of 0.001 is a good fitted model and it outperformed other experiments that were performed above using different optimizers and learning rates with an accuracy of 86.83%.

6.2 Evaluation

Not only the prediction of output summary by our model is important, but also, evaluating that result is also a significant task without which we cannot confirm whether the output produced by our model is efficient or not. Below are the quantitative and qualitative methods for evaluation of the generated summary.

6.2.1 Quantitative Analysis

ROUGE Metrics

In Text Summarization, summary evaluation is an essential chore. We use ROUGE metrics for our evaluation process. ROUGE refers to Recall Oriented Understudy for Gisting Evaluation (Lin; 2004) which is an automatic summary evaluation bench-marking metric that is widely used by researchers to determine the quality of the summary produced by comparing the machine generated summary with the reference summary (ideal or human written ones). ROUGE scores are computed from the number of overlapping words between the reference summary and machine generated summary.There are different types of ROUGE such as ROUGE-N, ROUGE-L, ROUGE-S and ROUGE-W. But the most commonly used ones are ROUGE-N *(ROUGE-1, ROUGE-2)* and ROUGE-L and hence we also use the same for our research.

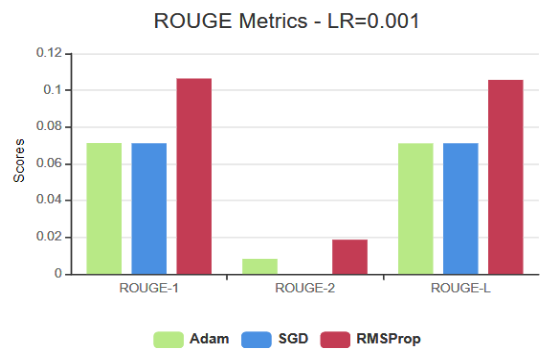
1. *ROUGE-N:* It denotes the overlapping of n-grams between the system generated summary and the ideal reference summary. For instance, unigram (ROUGE-1), bigram (ROUGE-2), trigram (ROUGE-3) and so on.
   The ROUGE-N is given by,

$$ROUGE - N = \frac{N_{(System,Reference)}}{n - gram_{\text{Reference}}} \tag{4}$$

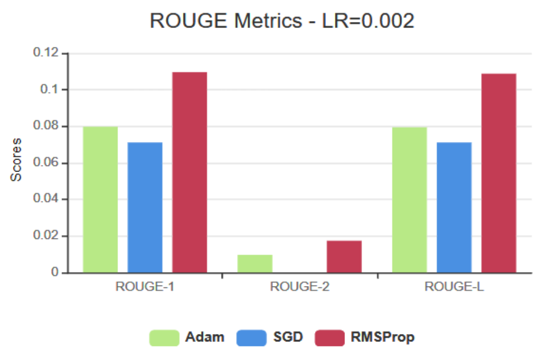
   where,
   $\text{N}_{\text{(System,Reference)}}$ - number of n-grams overlap between system summary and reference summary
   $\text{n-gram}_{\text{Reference}}$ - total number of n-grams in the reference summary

2. *ROUGE-L:* It denotes the Longest Common Subsequence (LCS) matching between the reference summary and system generated summary.

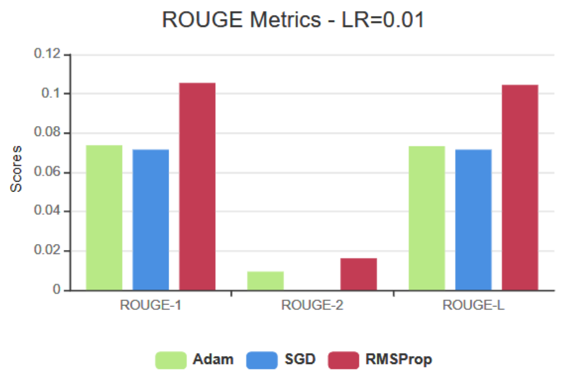
In our case, the dataset itself consists of actual summary which is the ideal reference summary that will be considered for the evaluation purpose. The ROUGE scores are calculated in every case which we considered during model training as shown in the Table 4 where R-1 is ROUGE-1, R-2 is ROUGE-2, R-L is ROUGE-L and LR is Learning Rate of the model.



(a) ROUGE Scores - LR=0.001



(b) ROUGE Scores LR=0.002



(c) ROUGE Scores LR=0.01

Figure 8: ROUGE Scores

Model Optimizer	**LR=0.001**			**LR=0.002**			**LR=0.01**		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Adam	7.1196	0.818	7.099	7.9733	0.9628	7.9438	7.3635	0.9282	7.3255
SGD	7.1024	0.0	7.1024	7.1035	0.0	7.1035	7.1424	0.0	7.1424
RMSProp	10.6304	1.8541	10.5664	10.9534	1.7315	10.8739	10.5389	1.6054	10.4386

Table 4: ROUGE Evaluation Scores

A graphical representation of the ROUGE scores for the output summaries generated in all the experiments that were performed on the model is shown in the Figure 8. It is evident from the bar graphs that in all the scenarios, the model with RMSProp optimizer has the highest R-1, R-2 and R-L scores. The good fit model with RMSProp optimizer at a learning rate of 0.001 which is concluded from the above experiments has the highest ROUGE scores as illustrated in figure 8(a). Thus, it is proved that the top performing model produced far better output summaries than the other models.

6.2.2 Qualitative Analysis

We also performed human evaluation of the predicted summaries in order to check whether the top performing model with highest ROUGE scores produced high quality more readable summaries. Five different people evaluated some 35 random samples from the validation dataset. They analysed the original review text, actual summary and system predicted summary and rated them into good, poor or moderate categories. The human evaluation results showed that nearly 75% of summaries were rated as *'Good'* whereas about 25% into *'Poor or Moderate'* ones. The below Table 4 shows few examples of both good and poor summaries generated by our best performing model.

1	**Original Review Text:** Cats are finicky animals and what one loves another won't touch - that is not a reflection on the food itself. I have two Burmese and their eating habits are very different. My local Pet Nutrition Center kindly gave me a sample of Prowl the other day and while my one cat absolutely loves it, the other won't touch it. That's why samples are nice. Both of my cats have been on Evo kibble, but this one cat who now loves the Prowl, has a habit of throwing up and I'm on the prowl myself for an alternative food that will help her keep food down. So far, no barf. She also laps it up as soon as I've added warm water - no waiting. I highly recommend you obtain a sample and let the cat decide, as the food is economical as well as healthy. **Actual Summary:** Economical and my cat loves it **Model Predicted Summary:** my cats love it **(Good)**
2	**Original Review Text:** I popped my first batch and I have to say this does taste pretty darn close to what you'd find at a movie theatre. Everything seems to be measured well enough that you don't see any grease stains when using a popcorn bag but you still get that distinct taste and flavor. The popcorn tasted fresh but there are no expiration dates or lot numbers anywhere on the packaging so I'm not sure how they can trace anything if there's a bad batch. They don't have any ISO 9001 markings so I'm assuming there is no quality oversight of these. <br/><br />I have zero real complaints about the popcorn or the taste, but I don't think they should add some sort of traceability. **Actual Summary:** Tasty popcorn **Model Predicted Summary:** great popcorn **(Good)**
3	**Original Review Text:** It was recently suggested to me I try going gluten-free to battle an autoimmune thyroid disorder. I bought the Gluten-Free Bisquick hoping the pancakes would at least be edible. The Bisquick has far exceeded my expectations! The pancakes are light, fluffy, and delicious! I definitely recommend this product! **Actual Summary:** Great Pancakes! **Model Predicted Summary:** great gluten free bread **(Good)**
4	**Original Review Text:** We just learned about Dilmah teas and thought we would try this one first. It is fabulous! Not a bitter drop in the pot. I certainly will serve this tea to my friends with pride. It is a Fair Trade tea which matters a great deal to us. Give it a try! **Actual Summary:** best tea ever **Model Predicted Summary:** great coffee **(Poor)**

Table 5: Human Evaluation - Model Predicted Summaries Example

7 Conclusion and Future Work

In this research, we implemented a novel automatic CEATS summarization model which combines the advantages of both extractive and abstractive techniques. Our model was evaluated using the online food reviews dataset from Amazon. From the experiments performed in section 6, the model performed well without getting affected by either overfitting or underfitting conditions at a learning rate of 0.001 using RMSProp optimizer with 300 hidden RNN units in each layer of the encoder-decoder networks. This model produced more concise and readable summaries using novel words or phrases with an accuracy of about 86.83% which is better when compared to the other model experiments performed above using different optimizers and learning rates. Also, the evaluation of the model predicted summary proved that the inference from the experimental results were correct and the summaries generated were more human readable and concise in nature. This system can be utilized to summarize the comments or reviews on the e-commerce websites by helping customers to quickly analyse the quality of the products and decide whether to buy the item or not. It also helps the companies to improve their product quality and enhance business by reviewing millions of user comments within a short span of time. The major challenge which was faced in this research was the lack of sufficient computational resource in the local machine to handle the size of the dataset which was overcome by using a cloud based platform *Google Colab*.

As a part of future work, this summarization concept can be extended to various other domains such as education, science, R&D, tele-medicine, financial research, insurance sector, legal document analysis, News headlines, search engine optimization, digital or social media marketing, chat bots, emails skimming, customer support or contact center industry in order to efficiently retrieve the information, thus reducing the user's reading time. In today's more competitive world, all these above use cases will improve the customer experience in real time and enhance various businesses making them much more profitable. Also, this model can be trained and validated using different datasets from multiple sources and results can be interpreted and compared with that of the state-of-art models using those datasets. The number of hidden layers and other hyper-parameters can also be tuned in order to check whether the accuracy is improved or not.

8 Acknowledgement

I would like to thank my project supervisor and guide Divyaa Manimaran Elango for her continued guidance, advice and suggestions throughout my research thesis. I would also like to acknowledge my friends and family who always supported me throughout my master's degree.

References

Alami, N., En-nahnahi, N., Ouatik, S. A. and Meknassi, M. (2018). Using unsupervised deep learning for automatic summarization of arabic documents, *Arabian Journal for Science and Engineering* **43**(12): 7803–7815.

Alguliev, R. M., Aliguliyev, R. M., Hajirahimova, M. S. and Mehdiyev, C. A. (2011). Mcmr: Maximum coverage and minimum redundant text summarization model, *Expert Systems with Applications* **38**(12): 14514–14522.

Alguliev, R. M., Aliguliyev, R. M. and Isazade, N. R. (2013). Multiple documents summarization based on evolutionary optimization algorithm, *Expert Systems with Applications* **40**(5): 1675–1689.

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

Anzanello, M. J. and Fogliatto, F. S. (2011). Learning curve models and applications: Literature review and research directions, *International Journal of Industrial Ergonomics* **41**(5): 573–583.

Azevedo, A. I. R. L. and Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview, *IADS-DM* .

Bahdanau, D., Cho, K. et al. (2014). Neural machine translation by jointly learning to align and translate. arxiv preprint arxiv: 1409.0473.

Barrera, A. and Verma, R. (2012). Combining syntax and semantics for automatic extractive single-document summarization, *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, pp. 366–377.

Baxendale, P. B. (1958). Machine-made index for technical literature—an experiment, *IBM Journal of research and development* **2**(4): 354–361.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine, *Computer networks and ISDN systems* **30**(1-7): 107–117.

Chen, F., Han, K. and Chen, G. (2002). An approach to sentence-selection-based text summarization, *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings.*, Vol. 1, IEEE, pp. 489–493.

Chopra, S., Auli, M. and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98.

Edmundson, H. P. (1969). New methods in automatic extracting, *Journal of the ACM (JACM)* **16**(2): 264–285.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization, *Journal of artificial intelligence research* **22**: 457–479.

Fan, A., Grangier, D. and Auli, M. (2017). Controllable abstractive summarization, *arXiv preprint arXiv:1711.05217* .

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data, *Communications of the ACM* **39**(11): 27–34.

Gupta, S. and Gupta, S. (2018). Abstractive summarization: an overview of the state of the art, *Expert Systems with Applications* .

Hardeniya, N., Perkins, J., Chopra, D., Joshi, N. and Mathur, I. (2016). *Natural Language Processing: Python and NLTK*, Packt Publishing Ltd.

Hingu, D., Shah, D. and Udmale, S. S. (2015). Automatic text summarization of wikipedia articles, *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, IEEE, pp. 1–4.

Hou, L., Hu, P. and Bei, C. (2017). Abstractive document summarization via neural model with joint attention, *National CCF Conference on Natural Language Processing and Chinese Computing*, Springer, pp. 329–338.

Hsu, W.-T., Lin, C.-K., Lee, M.-Y., Min, K., Tang, J. and Sun, M. (2018). A unified model for extractive and abstractive summarization using inconsistency loss, *arXiv preprint arXiv:1805.06266* .

Ian, G. and Yoshua, B. (2016). Deep learning (adaptive computation and machine learning).

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An introduction to statistical learning*, Vol. 112, Springer.

Li, J., Huang, G., Fan, C., Sun, Z. and Zhu, H. (2019). Key word extraction for short text via word2vec, doc2vec, and textrank, *Turkish Journal of Electrical Engineering & Computer Sciences* **27**(3): 1794–1805.

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

Luhn, H. P. (1958). The automatic creation of literature abstracts, *IBM Journal of research and development* **2**(2): 159–165.

McAuley, J. J. and Leskovec, J. (2013). From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews, *Proceedings of the 22nd international conference on World Wide Web*, ACM, pp. 897–908.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text, *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411.

Muthiah, K. (2019). Automatic coherent and concise text summarization using nlp, *My Research In Computing at National College of Ireland* .

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B. et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond, *arXiv preprint arXiv:1602.06023* .

Qaiser, S. and Ali, R. (2018). Text mining: use of tf-idf to examine the relevance of words to documents, *International Journal of Computer Applications* **181**(1): 25–29.

Radev, D. R., Hovy, E. and McKeown, K. (2002). Introduction to the special issue on summarization, *Computational linguistics* **28**(4): 399–408.

Saggion, H. and Poibeau, T. (2013). Automatic text summarization: Past, present and future, *Multi-source, multilingual information extraction and summarization*, Springer, pp. 3–21.

Saziyabegum, S. and Sajja, P. S. (2016). Literature review on extractive text summarization approaches, *International Journal of Computer Applications* **156**(12).

See, A., Liu, P. J. and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks, *arXiv preprint arXiv:1704.04368* .

Sethi, P., Sonawane, S., Khanwalker, S. and Keskar, R. (2017). Automatic text summarization of news articles, *2017 International Conference on Big Data, IoT and Data Science (BID)*, IEEE, pp. 23–29.

Subramanian, S., Li, R., Pilault, J. and Pal, C. (2019). On extractive and abstractive neural document summarization with transformer language models, *arXiv preprint arXiv:1909.03186* .

Vanderwende, L., Suzuki, H., Brockett, C. and Nenkova, A. (2007). Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion, *Information Processing & Management* **43**(6): 1606–1618.

Yih, W.-t., Goodman, J., Vanderwende, L. and Suzuki, H. (2007). Multi-document summarization by maximizing informative content-words., *IJCAI*, Vol. 7, pp. 1776–1782.

Yousefi-Azar, M. and Hamey, L. (2017). Text summarization using unsupervised deep learning, *Expert Systems with Applications* **68**: 93–105.

Zhou, Q., Yang, N., Wei, F. and Zhou, M. (2017). Selective encoding for abstractive sentence summarization, *arXiv preprint arXiv:1704.07073* .

Configuration Manual

MSc Research Project
Cloud Computing

Kaarthic Muthiah
Student ID: X18129579

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

Configuration Manual

Kaarthic Muthiah
X18129579

1 Introduction

This configuration manual presents various steps or procedures involved in the implementation of our research project. It provides various details such as system specification, dataset collection and preparation, development of our CEATS model, experimental results and evaluation with appropriate URLs, code snippets and screenshots.

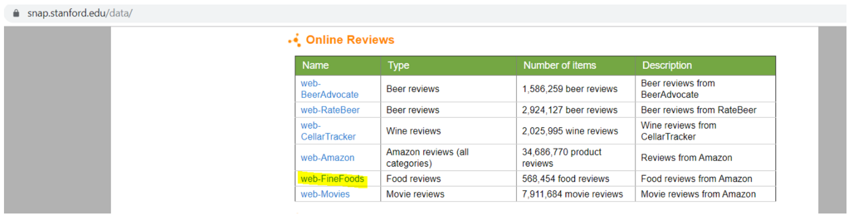
2 Pre-requisites and System Configuration

- **Programming Language:** Python (version 3.6.9)
- **IDE:** Google Colab - Cloud-based Jupyter notebook environment
- **Computation:** Google Compute Engine backend - free tier google account with free GPU - 1XTesla K80, 2496 CUDA cores or free TPU - TPU v2, 8 cores, approx 12 GB RAM with a maximum RAM limit up to 36 GB
- **Browser:** Google Chrome

3 Dataset Preparation

3.1 Collecting Online Food Reviews

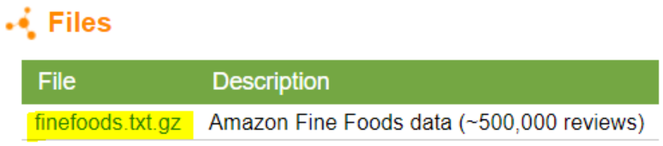
1. Open the URL `https://snap.stanford.edu/data/` to reach the SNAP Stanford dataset repository.
2. Go to the Online Reviews section and click on 'web-FineFoods' to download the *'fine-foods.txt.gz'* zip file.
3. Extract the zip file and you will get a text file *'foods.txt'*.



Online Reviews

Name	Type	Number of items	Description
web-BeerAdvocate	Beer reviews	1,586,259 beer reviews	Beer reviews from BeerAdvocate
web-RateBeer	Beer reviews	2,924,127 beer reviews	Beer reviews from RateBeer
web-CellarTracker	Wine reviews	2,025,995 wine reviews	Wine reviews from CellarTracker
web-Amazon	Amazon reviews (all categories)	34,686,770 product reviews	Reviews from Amazon
web-FineFoods	Food reviews	568,454 food reviews	Food reviews from Amazon
web-Movies	Movie reviews	7,911,684 movie reviews	Movie reviews from Amazon

Figure 1: SNAP Stanford Online Reviews Repository



Files

File	Description
finefoods.txt.gz	Amazon Fine Foods data (~500,000 reviews)

Figure 2: Fine Food Reviews Zip file

4. This text file can be uploaded to the google drive so that it can be accessed easily for future use in the following steps.

3.2 Parsing Food Reviews

1. Now sign in to Google Colab free tier account (`https://colab.research.google.com/`) using your Google email id.
2. Navigate to File -> New Python 3 notebook and create a new jupyter notebook file. Rename the file accordingly by double clicking on the file name (Figure 3).

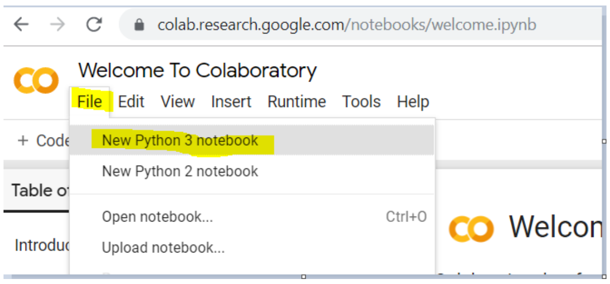
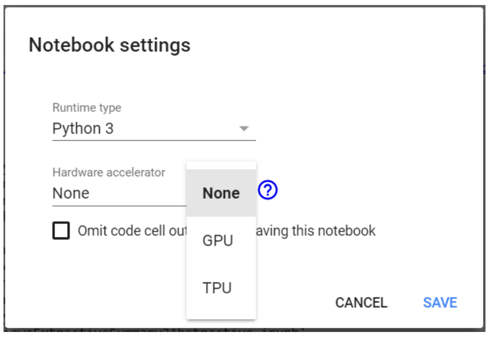


Figure 3: Google Colab - New Python Notebook

3. Next step is to set up the runtime type (free GPU or TPU) by clicking Runtime -> Change runtime type. Then, select either GPU or TPU from the *'Hardware accelarator'* drop-down menu (Figure 4).



Notebook settings

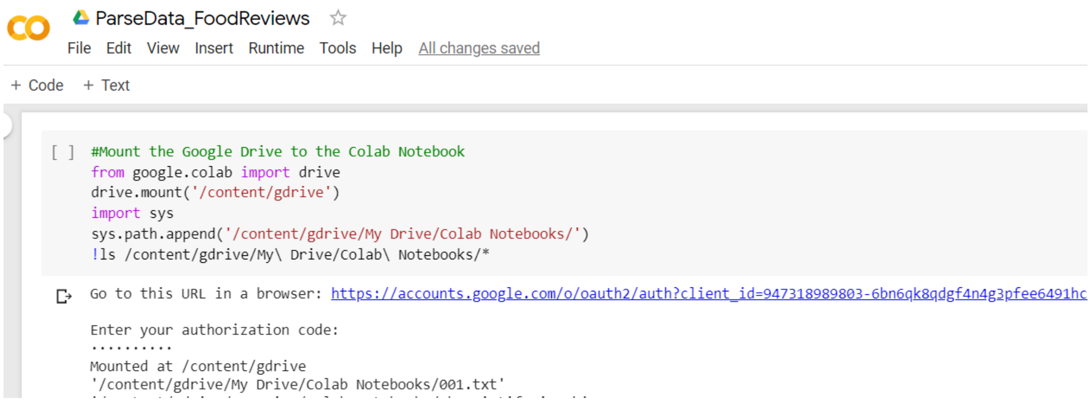
Runtime type: Python 3

Hardware accelerator: None

Figure 4: Hardware Accelerator

4. Mount the Google Drive to your colab notebook by running the code shown in Figure 5. It will show a link, click on that link to get the authentication code to mount

your drive and access it.

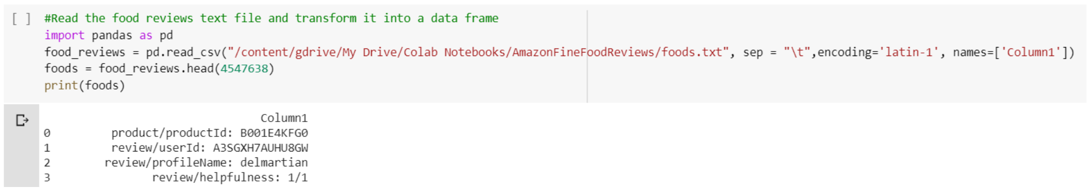


```
[ ] #Mount the Google Drive to the Colab Notebook
    from google.colab import drive
    drive.mount('/content/gdrive')
    import sys
    sys.path.append('/content/gdrive/My Drive/Colab Notebooks/')
    !ls /content/gdrive/My\ Drive/Colab\ Notebooks/*
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc

Enter your authorization code:
··········
Mounted at /content/gdrive
'/content/gdrive/My Drive/Colab Notebooks/001.txt'
```

Figure 5: Mounting Google Drive

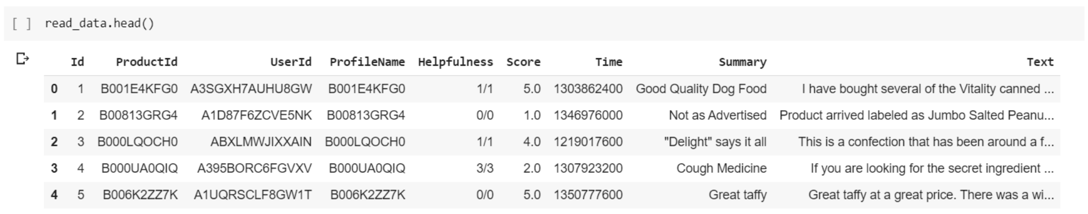
5. The code for parsing the food reviews text file and converting into the dataset CSV file is given in 'ParseData_FoodReviews.ipynb'. Initially the text file is transformed into a pandas dataframe as shown in figure 6. Next after following the code given in the above python notebook file, the final dataset CSV produced will look like the one showed in figure 7.



```
[ ] #Read the food reviews text file and transform it into a data frame
    import pandas as pd
    food_reviews = pd.read_csv("/content/gdrive/My Drive/Colab Notebooks/AmazonFineFoodReviews/foods.txt", sep = "\t",encoding='latin-1', names=['Column1'])
    foods = food_reviews.head(4547638)
    print(foods)
```

```
                                Column1
0          product/productId: B001E4KFG0
1           review/userId: A3SGXH7AUHU8GW
2         review/profileName: delmartian
3               review/helpfulness: 1/1
```

Figure 6: Text file to Pandas DataFrame



```
[ ] read_data.head()
```

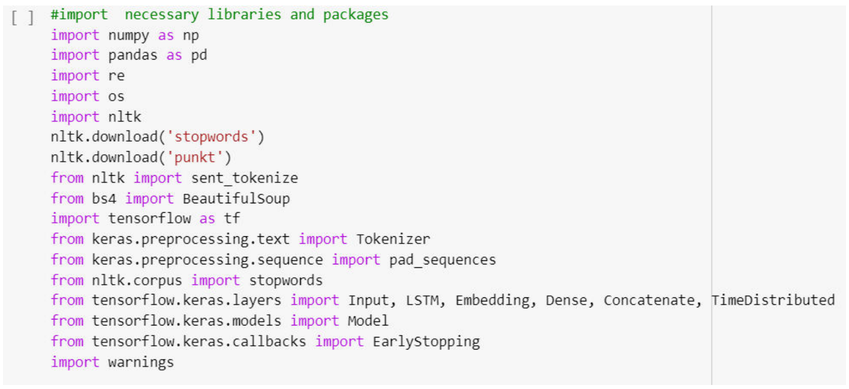
	Id	ProductId	UserId	ProfileName	Helpfulness	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	B001E4KFG0	1/1	5.0	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned ...
1	2	B00813GRG4	A1D87F6ZCVE5NK	B00813GRG4	0/0	1.0	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanu...
2	3	B000LQOCH0	ABXLMWJIXXAIN	B000LQOCH0	1/1	4.0	1219017600	"Delight" says it all	This is a confection that has been around a f...
3	4	B000UA0QIQ	A395BORC6FGVXV	B000UA0QIQ	3/3	2.0	1307923200	Cough Medicine	If you are looking for the secret ingredient ...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	B006K2ZZ7K	0/0	5.0	1350777600	Great taffy	Great taffy at a great price. There was a wi...

Figure 7: Dataset CSV

4 CEATS Model - Project Development

Once the dataset CSV 'ParsedFoodReviews.csv' is created, it is uploaded to the google drive to be used in our model development stage. There are different attributes in the dataset namely Id, ProductId, UserID, Username, Ratings, Helpfulness ratio, Time, Text and Reference Summary. For our purpose of text summarization, we use only two of these fields - Text and Summary. From approximately 500,000 food reviews, we choose data samples consisting of about 100,000 reviews.

4.1 Libraries necessary for implementation

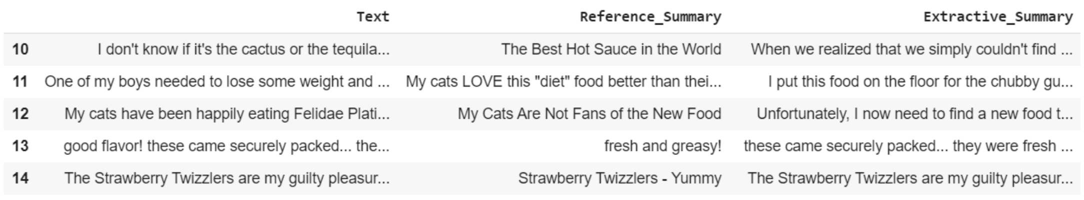


```
[ ] #import  necessary libraries and packages
    import numpy as np
    import pandas as pd
    import re
    import os
    import nltk
    nltk.download('stopwords')
    nltk.download('punkt')
    from nltk import sent_tokenize
    from bs4 import BeautifulSoup
    import tensorflow as tf
    from 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
```

Figure 8: Libraries required for developing our model

4.2 Extractive Summarization

The first phase of our model is extractive summarization which is used to pick the key sentences from the input food review text. The major steps involved in this are data cleaning, feature extraction and sentence ranking. The complete code for this phase is in 'Phase1_Extractive.ipynb' file. Once the important points are extracted from the review text, it is transformed into a CSV file - 'FoodReviewsExtractiveSummary.csv'. It consists of three columns namely Text, Reference_Summary and Extractive_Summary as shown in figure 9.

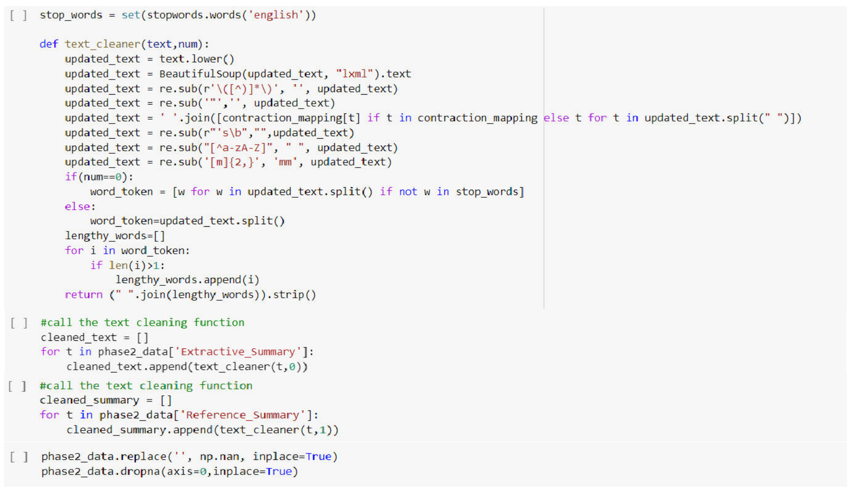


	Text	Reference_Summary	Extractive_Summary
10	I don't know if it's the cactus or the tequila...	The Best Hot Sauce in the World	When we realized that we simply couldn't find ...
11	One of my boys needed to lose some weight and ...	My cats LOVE this "diet" food better than thei...	I put this food on the floor for the chubby gu...
12	My cats have been happily eating Felidae Plati...	My Cats Are Not Fans of the New Food	Unfortunately, I now need to find a new food t...
13	good flavor! these came securely packed... the...	fresh and greasy!	these came securely packed... they were fresh ...
14	The Strawberry Twizzlers are my guilty pleasur...	Strawberry Twizzlers - Yummy	The Strawberry Twizzlers are my guilty pleasur...

Figure 9: Extractive Summary - phase 1 output CSV file

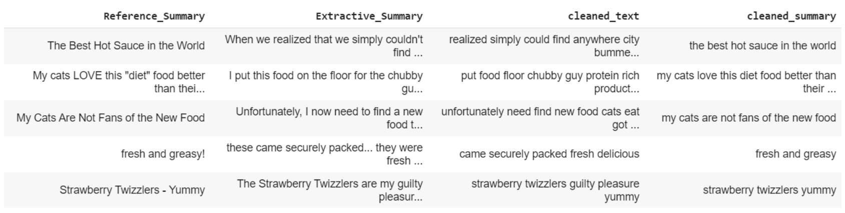
4.3 Abstractive Summarization

In this phase, we select 50000 samples from the phase 1 output as input for this abstractive phase. As the first step, we need to clean the review text and summary being fed into the deep learning model. The text pre-processing is followed in a similar way as the first stage which is shown in figure 10.



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

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

[ ]  #call the text cleaning function
     cleaned_text = []
     for t in phase2_data['Extractive_Summary']:
         cleaned_text.append(text_cleaner(t,0))

[ ]  #call the text cleaning function
     cleaned_summary = []
     for t in phase2_data['Reference_Summary']:
         cleaned_summary.append(text_cleaner(t,1))

[ ]  phase2_data.replace('', np.nan, inplace=True)
     phase2_data.dropna(axis=0,inplace=True)
```

Figure 10: Abstractive Phase - Data Cleaning Code snippet



Reference_Summary	Extractive_Summary	cleaned_text	cleaned_summary
The Best Hot Sauce in the World	When we realized that we simply couldn't find ...	realized simply could find anywhere city bumme...	the best hot sauce in the world
My cats LOVE this "diet" food better than thei...	I put this food on the floor for the chubby gu...	put food floor chubby guy protein rich product...	my cats love this diet food better than their ...
My Cats Are Not Fans of the New Food	Unfortunately, I now need to find a new food t...	unfortunately need find new food cats eat got ...	my cats are not fans of the new food
fresh and greasy!	these came securely packed... they were fresh ...	came securely packed fresh delicious	fresh and greasy
Strawberry Twizzlers - Yummy	The Strawberry Twizzlers are my guilty pleasur...	strawberry twizzlers guilty pleasure yummy	strawberry twizzlers yummy

Figure 11: Phase 2 - Cleaned input extractive summary and cleaned reference summary

After cleaning the data, that is, the extractive summary which is the input text to this phase and the actual reference summary appears like what is shown in figure 11. Now the distribution of lengths of both cleaned text and cleaned summary are plotted with help of 'matplotlib' library in python as shown in figure 12. Using this histogram, we can fix the maximum lengths of the review text and summaries.

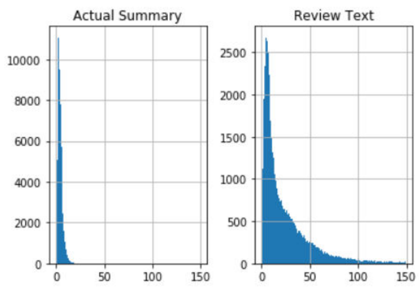
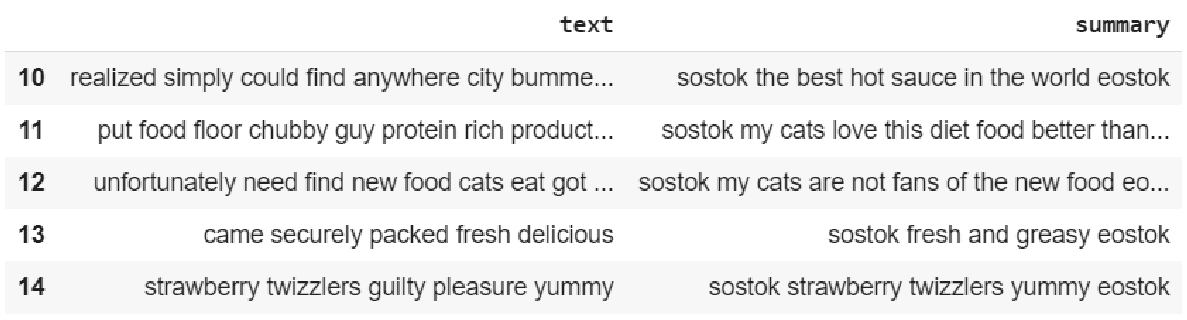


Figure 12: Distribution of text and summary lengths

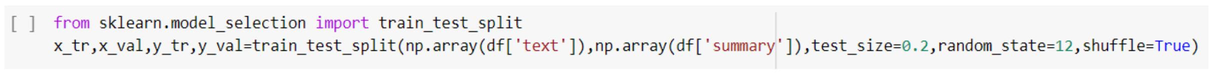
Then we add the start and end tokens namely 'sostok' and 'eostok' to the summary before feeding into the model, in order to easily identify the starting and ending of the sequence as presented in figure 13.

Now, as shown in figure 14, the dataset is splitted in the 80:20 ratio for training and validation purposes respectively.Then, we build a tokenizer based on the most commonly used words in the review and summary to create a vocabulary of unique words and convert word to integer sequences.



	text	summary
10	realized simply could find anywhere city bumme...	sostok the best hot sauce in the world eostok
11	put food floor chubby guy protein rich product...	sostok my cats love this diet food better than...
12	unfortunately need find new food cats eat got ...	sostok my cats are not fans of the new food eo...
13	came securely packed fresh delicious	sostok fresh and greasy eostok
14	strawberry twizzlers guilty pleasure yummy	sostok strawberry twizzlers yummy eostok

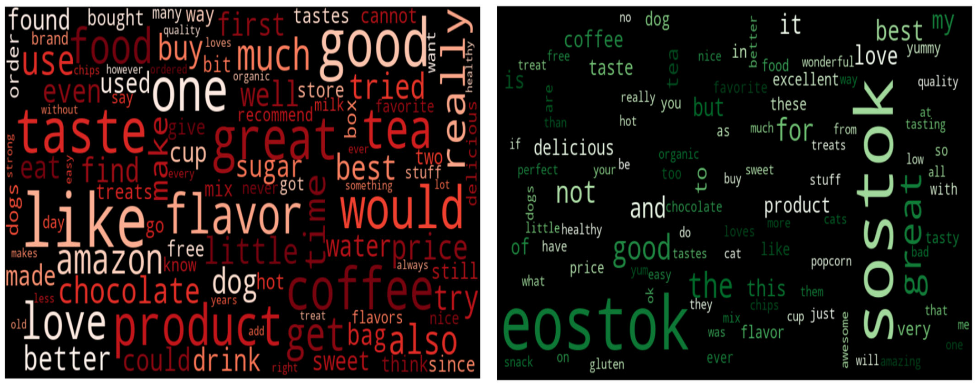
Figure 13: Adding Start and End tokens to the summary



```
[ ] from sklearn.model_selection import train_test_split
    x_tr,x_val,y_tr,y_val=train_test_split(np.array(df['text']),np.array(df['summary']),test_size=0.2,random_state=12,shuffle=True)
```

Figure 14: Splitting Dataset - 80:20 ratio

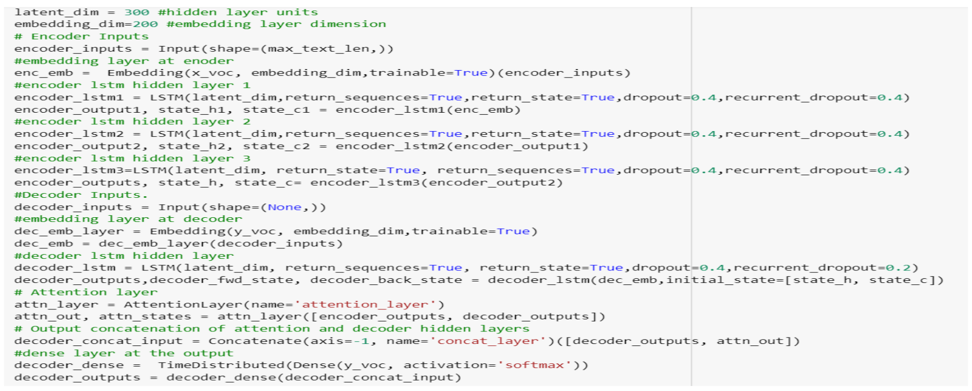
A word cloud of the most common words in reviews text and summary are shown in figure 15.



(a) Review Text (b) Summary

Figure 15: Word Cloud - Review Text Summary

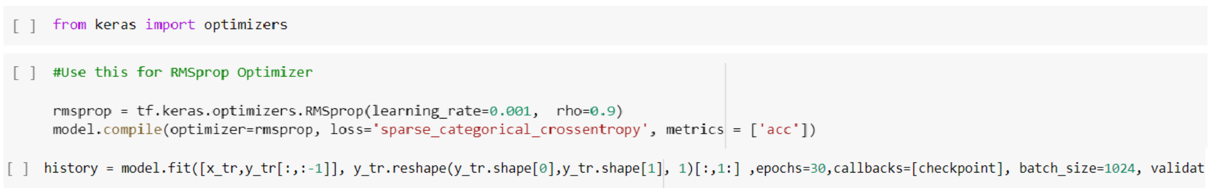
The three layer LSTM encoder, single layer decoder LSTM recurrent neural network model built in this research can be seen in the figure 16. The various optimizer functions (Adam, RMSProp and SGD) are imported from the 'keras' machine learning library. The model is then compiled by using the optimizer function. After compiling, the model is trained using 'model.fit' function shown in figure 17.



```
latent_dim = 300 #hidden layer units
embedding_dim=200 #embedding layer dimension
# Encoder Inputs
encoder_inputs = Input(shape=(max_text_len,))
#embedding layer at enoder
enc_emb =  Embedding(x_voc, embedding_dim,trainable=True)(encoder_inputs)
#encoder lstm hidden layer 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 hidden layer 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 hidden layer 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)
#Decoder Inputs.
decoder_inputs = Input(shape=(None,))
#embedding layer at decoder
dec_emb_layer = Embedding(y_voc, embedding_dim,trainable=True)
dec_emb = dec_emb_layer(decoder_inputs)
#decoder lstm hidden layer
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])
# Output concatenation of attention and decoder hidden layers
decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])
#dense layer at the output
decoder_dense =  TimeDistributed(Dense(y_voc, activation='softmax'))
decoder_outputs = decoder_dense(decoder_concat_input)
```

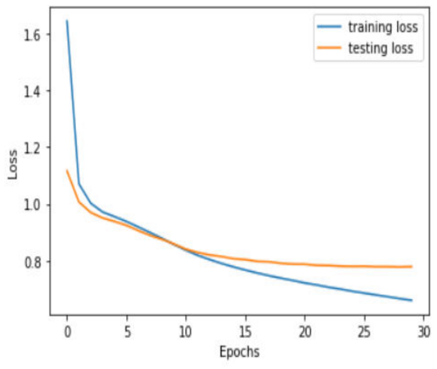
Figure 16: Recurrent Neural Network Model

After model is trained, say for about 30 epochs until it shows good fit behaviour without getting overfitted, the loss and accuracy learning curves are plotted for both training and validation dataset. It is shown in figure 18. The code for this phase is given in 'Phase2FoodReviewsAbstractiveSummarizer.ipynb' file.

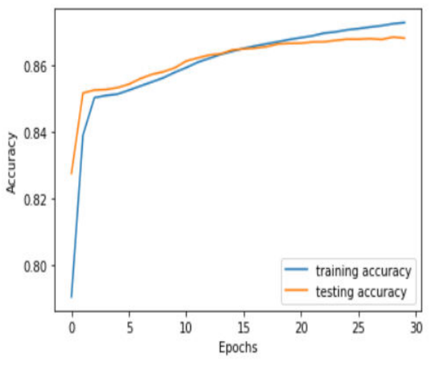


```
[ ] from keras import optimizers

[ ] #Use this for RMSprop Optimizer

    rmsprop = tf.keras.optimizers.RMSprop(learning_rate=0.001,  rho=0.9)
    model.compile(optimizer=rmsprop, loss='sparse_categorical_crossentropy', metrics = ['acc'])

[ ] history = model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=30,callbacks=[checkpoint], batch_size=1024, validat
```

Figure 17: Model Compiling and Training



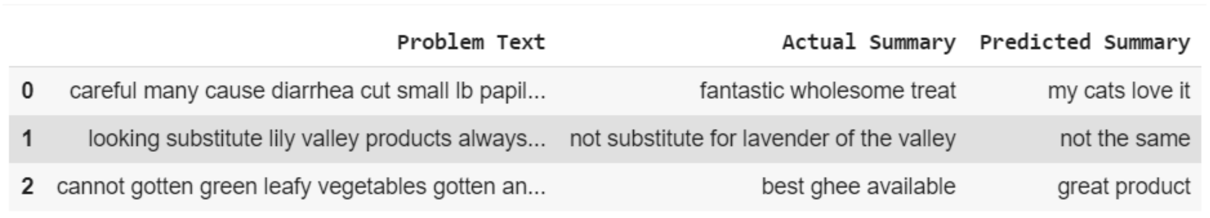
(a) Loss Curve



(b) Accuracy Curve

Figure 18: Learning Curves for our LSTM RNN encoder decoder model

Finally, the samples from validation dataset will be passed into the model and output summary will be predicted by the model. The final output summary is stored in 'FoodReviewsFinalPredictedSummary.csv' file. A smaple predicted summary is shown in figure 19



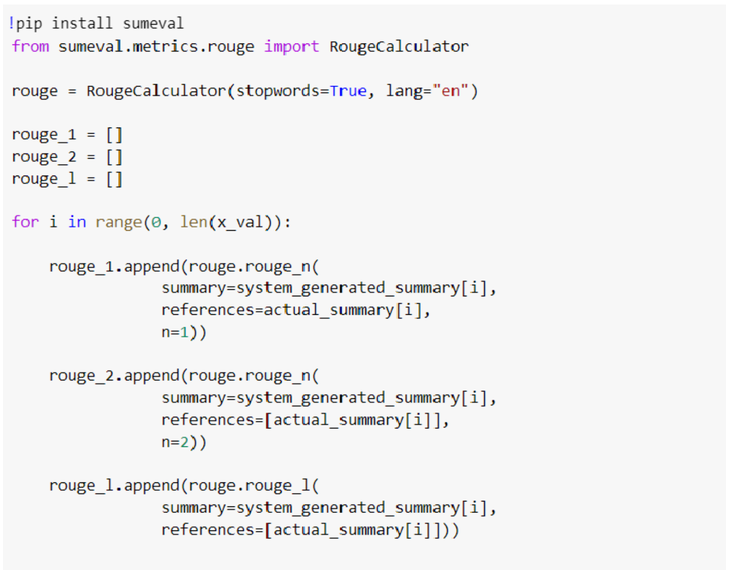
	Problem Text	Actual Summary	Predicted Summary
0	careful many cause diarrhea cut small lb papil...	fantastic wholesome treat	my cats love it
1	looking substitute lily valley products always...	not substitute for lavender of the valley	not the same
2	cannot gotten green leafy vegetables gotten an...	best ghee available	great product

Figure 19: Sample of Final Predicted Summary

5 Evaluation

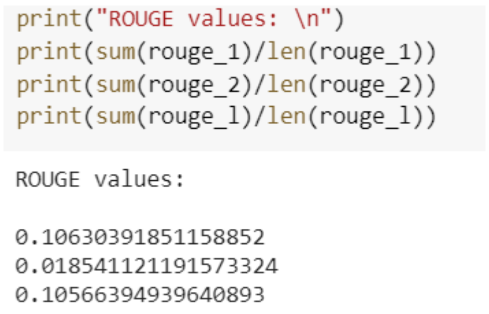
5.1 ROUGE Metrics

For a quantitative analysis of the predicted summary, we use ROUGE evaluation metrics and the scores are calculated.



```
!pip install sumeval
from sumeval.metrics.rouge import RougeCalculator

rouge = RougeCalculator(stopwords=True, lang="en")

rouge_1 = []
rouge_2 = []
rouge_l = []

for i in range(0, len(x_val)):

    rouge_1.append(rouge.rouge_n(
              summary=system_generated_summary[i],
              references=actual_summary[i],
              n=1))

    rouge_2.append(rouge.rouge_n(
              summary=system_generated_summary[i],
              references=[actual_summary[i]],
              n=2))

    rouge_l.append(rouge.rouge_l(
              summary=system_generated_summary[i],
              references=[actual_summary[i]]))
```

Figure 20: ROUGE score Calculation



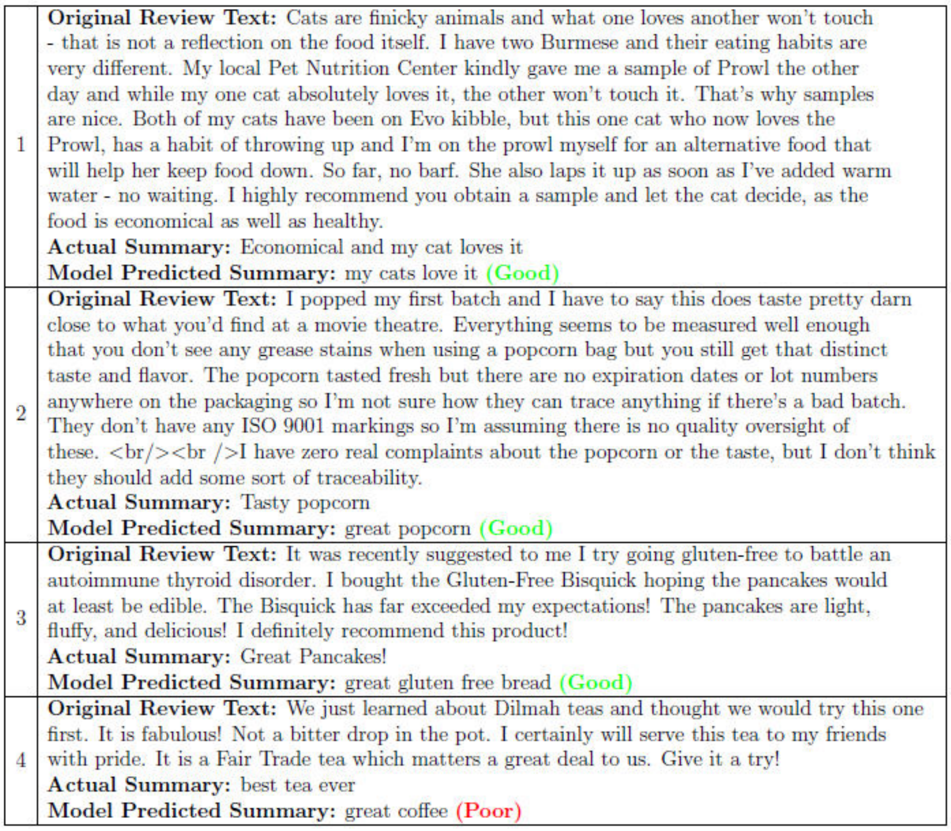
```
print("ROUGE values: \n")
print(sum(rouge_1)/len(rouge_1))
print(sum(rouge_2)/len(rouge_2))
print(sum(rouge_l)/len(rouge_l))
```

```
ROUGE values:

0.10630391851158852
0.018541121191573324
0.10566394939640893
```

Figure 21: ROUGE-1, ROUGE-2, ROUGE-L scores

5.2 Human Evaluation

The qualitative analysis of our model predicted summary was done by human evaluators by comparing the original review text, actual summary and predicted summary. The figure 22 shows a sample of good and poor summaries.



1	**Original Review Text:** Cats are finicky animals and what one loves another won't touch - that is not a reflection on the food itself. I have two Burmese and their eating habits are very different. My local Pet Nutrition Center kindly gave me a sample of Prowl the other day and while my one cat absolutely loves it, the other won't touch it. That's why samples are nice. Both of my cats have been on Evo kibble, but this one cat who now loves the Prowl, has a habit of throwing up and I'm on the prowl myself for an alternative food that will help her keep food down. So far, no barf. She also laps it up as soon as I've added warm water - no waiting. I highly recommend you obtain a sample and let the cat decide, as the food is economical as well as healthy. **Actual Summary:** Economical and my cat loves it **Model Predicted Summary:** my cats love it (Good)
2	**Original Review Text:** I popped my first batch and I have to say this does taste pretty darn close to what you'd find at a movie theatre. Everything seems to be measured well enough that you don't see any grease stains when using a popcorn bag but you still get that distinct taste and flavor. The popcorn tasted fresh but there are no expiration dates or lot numbers anywhere on the packaging so I'm not sure how they can trace anything if there's a bad batch. They don't have any ISO 9001 markings so I'm assuming there is no quality oversight of these. <br/><br />I have zero real complaints about the popcorn or the taste, but I don't think they should add some sort of traceability. **Actual Summary:** Tasty popcorn **Model Predicted Summary:** great popcorn (Good)
3	**Original Review Text:** It was recently suggested to me I try going gluten-free to battle an autoimmune thyroid disorder. I bought the Gluten-Free Bisquick hoping the pancakes would at least be edible. The Bisquick has far exceeded my expectations! The pancakes are light, fluffy, and delicious! I definitely recommend this product! **Actual Summary:** Great Pancakes! **Model Predicted Summary:** great gluten free bread (Good)
4	**Original Review Text:** We just learned about Dilmah teas and thought we would try this one first. It is fabulous! Not a bitter drop in the pot. I certainly will serve this tea to my friends with pride. It is a Fair Trade tea which matters a great deal to us. Give it a try! **Actual Summary:** best tea ever **Model Predicted Summary:** great coffee (Poor)

Figure 22: Human Evaluation - Sample of Good and Poor Summaries