ABSTRACTIVE TEXT SUMMARIZATION

A PROJECT REPORT

Submitted by

M.RAAHAVI (1605109)

M.REHAPRIADARSINI (1605117)

S.S. SUDARSHANA (1605141)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SRI RAMAKRISHNA ENGINEERING COLLEGE

[Educational Service: SNR Sons Charitable Trust]
[Autonomous Institution, Accredited by NAAC with 'A' Grade]
[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]
[ISO 9001:2015 Certified and All Eligible Programmes Accredited by NBA]
Vattamalaipalayam, N.G.G.O. Colony Post,

COIMBATORE – 641 022

ANNA UNIVERSITY: CHENNAI -600 025

JULY 2020

SRI RAMAKRISHNA ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Department of Information Technology

PROJECT WORK - JULY 2020

This is to certify that the project entitled

ABSTRACTIVE TEXT SUMMARIZATION

is the bonafide record of project work done by

M.RAAHAVI	**(1605109)**
M.REHAPRIADARSINI	**(1605117)**
S.S.SUDARSHANA	**(1605141)**

who carried out the project work under my supervision, certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

of B.Tech Information Technology during the year 2016-2020.

---------- ----------

Head of the Department Guide

Submitted for the Project Viva-Voce examination held on 15/07/2020

---------- ---------- ----------

Internal Examiner Subject Expert Subject Expert

ACKNOWLEDGEMENT

We thank the Almighty for his blessings on us to complete this project work successfully.

With profound sense of gratitude we sincerely thank the Management, **Managing Trustee**, **Thiru D.Lakshminarayanaswamy** and **Joint Managing Trustee**, **Thiru R.Sundar,** for having provided us the necessary infrastructure required for the completion of our project.

With profound sense of gratitude, we sincerely thank the **Head of the Institution**, **Dr.N.R.Alamelu** for her kind patronage, which helped in pursuing the project successfully.

With immense pleasure, we express our hearty thanks to **Head of the Department**, **Dr.M.Senthamil Selvi,** for her encouragement towards the completion of this project.

With immense pleasure, we thank our **Project co-ordinator** and **Project guide, Dr.J.Anitha, Associate Professor**, Department of Information Technology for her encouragement, valuable guidance and keen interest towards the completion of the project.

We convey our thanks to all the teaching and non-teaching staff members of our department who rendered their co-operation by all means for completion of this project.

ABSTRACT

Text summarization is the process of retrieving important information from a massively available data. The purpose of text summarization is that it reduces time, makes the selection process easier, minimizes the number of references and creates a cohesive and readable summary. Abstractive summarization is the method of generating a summary from its main key facts or ideas, not by replicating verbatim most salient sentences from text. The system uses a recurrent attentive summarizer model with ConceptNet Numberbatch word embedding to establish summaries from the context. The model uses a standard dataset entitled News_Summary. For a perfect abstractive summary, the model has to be first trained in such way that it truly understands the document and try to express that understanding possibly in a precise way using new phrases and words. The Long Short-Term Memory algorithm are used to produce efficient summaries correspondence to loaded word embeddings. The experimental results show a Precision of 0.4, Recall of 0.705, ROUGE score of 0.51.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
NLP	Natural Language Processing
HITS	Hyperlink-Induced Topic Search
TF	Term Frequency
ISF	Inverse Sentence Frequency
AMR	Abstract Meaning Representation
RBM	Restricted Boltzmann Machine
RSG	Rich Semantic Graph
LSTM	Long Short-Term Memory
NN	Neural Network
RNN	Recurrent Neural Network
NLTK	Natural Language ToolKit

CHAPTER 1

INTRODUCTION

Today, a tremendous amount of data is available on the Internet. With the proliferation of the Internet, it has become increasingly difficult to efficiently reach information from such massive amounts of data. This issue is not only related to the volume of the data, but also to the diversity of subjects, qualities, and idioms. It is a complex and exhaustive process to gather and perceive any main information from such a massive amount of resources in sufficient time for humans. Fortunately, these processes can be automatically performed by information retrieval methods. However, the rising quantity of information causes some performance issues, such as insufficient solutions and unwieldy applications of information retrieval tasks. The use of higher-technology machines may reduce the losses caused by these issues. However, they may cost more.

As a more applicable alternative, dimension reduction can be employed in raw data to handle these issues and to accelerate the implementations of these tasks. As regards the domain of text processing, automatic text summarization is a good practice for dimension reduction.[1] Text summarization is the process of generating a brief version of a single document or document set, so that the resulting summaries preserve the fundamental information of the original documents. This study addresses two motivations regarding automatic summary generation: summarization for machine, and summarization for human. First, insofar as information retrieval tasks, a reduction approach is satisfactory if it can be applied efficiently. This addresses summarization for machine, and more specifically, the use of summarization as a dimension reduction for information retrieval purposes.[2]

1.1. DEEP LEARNING

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.[3]

1.2. NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing is the technology used to aid computers to understand the human's natural language. Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable.

Syntax and semantic analysis are two main techniques used with natural language processing. Syntax is the arrangement of words in a sentence to make grammatical sense. NLP uses syntax to assess meaning from a language based on grammatical rules. Syntax techniques used include parsing (grammatical analysis for a sentence), word segmentation (which divides a large piece of text to units), sentence breaking (which places sentence boundaries in large texts),

morphological segmentation (which divides words into groups) and stemming (which divides words with inflection in them to root forms). Semantics involves the use and meaning behind words. NLP applies algorithms to understand the meaning and structure of sentences. Techniques that NLP uses with semantics include word sense disambiguation (which derives meaning of a word based on context), named entity recognition (which determines words that can be categorized into groups), and natural language generation (which will use a database to determine semantics behind words).[4]

1.3. TEXT SUMMARIZATION

Text summarization refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document. Automatic summarization is the process of shortening a set of data computationally, to create a subset that represents the most important or relevant information within the original content.[5]

1.4. TEXT SUMMARIZATION APPROACHES

There are two approaches in Text Summarization, namely: Abstractive Text Summarization and Extractive Text Summarization.

1.4.1. Extractive Text Summarization

Extractive method is choosing specific main words from the input to generate the output, this model tends to work, but won't output correctly structured sentences, as it just selects words from input and copy them to the output, without actually understanding the sentences, as a highlighter which selects the main information from a source text. In machine learning, extractive summarization usually involves weighing the essential sections of sentences and using the results to generate summaries. Different types of algorithms and methods can be used to gauge the weights of the sentences and then rank them

according to their relevance and similarity with one another and further joining them to generate a summary.[6]

1.4.2. Abstractive Text Summarization

Abstractive summarization is the technique of generating a summary of a text from its main ideas, not by copying verbatim most salient sentences from text. This is an important and challenge task in natural language processing. Abstractive text summarization is based on discourse rules, syntactic constraints and word graph. Discourse rules and syntactic constraints are used in the process of generating sentences from keywords. Word graph is used in the sentence combination process to represent word relations in the text and to combine several sentences into one. Abstractive method transforms the aforementioned generalized summary into human-readable form.[7]

CHAPTER 2

LITERATURE SURVEY

Automatic text summarization aims to reduce the document text size by building a brief and voluble summary that has the most important ideas in that document. Through the years, many approaches were proposed to improve the automatic text summarization results; the graph-based method for sentence ranking is considered one of the most important approaches in this field.

2.1 Graph based Multi-document Summarization [8]

The proposed system is based on two of the most important work on Graph-based methods for sentence ranking; TextRank and LexRank. It provides an improved weighted scheme by combining multiple important measures that calculate the similarity between two sentences, which are: Jaccard similarity coefficient, TF*IDF cosine similarity, Topic signature similarity and the identity similarity measure. To combine these measures, they have experimented two different ways of results averaging, which are: arithmetic mean and harmonic mean. In case of multi-document summarization PageRank algorithm is used to rank the webpages, where-in it computes the ranking score for each vertex in the graph based on the probability of being in that vertex.

The model uses a new ranking technique to rank the graph vertices in which it involves harmonic mean to combine the results of two most important graph-based methods: PageRank and HITS algorithm. In addition, it uses straight forward approach to extract the summaries through simple steps that do not require complex linguistic processing or labeled training data. The drawback is the proposal of taking the average ranking scores using the harmonic mean obtained comparable results to the PageRank and did not give the desired improvement. For efficient results more advanced methods for combining scores like machine learning techniques to learn what is the best score among that can be used among all proposed scores should be implemented.

2.2 Multi-document extractive text summarization [9]

The study aims to generate informative text summaries that substitute for the original document as much as possible. The problem was considered as sentence extraction from multiple documents on a subject for summarization purposes. Particularly, the best feature vector was investigated based on considering what features should be used in which way, as existing studies calculated feature values by numerous methods, but did not determine a standard on how to measure those values. The model evaluation was performed based on both standard classification performance metrics such as precision, recall, accuracy, and ROC-AUC, and by ROUGE analysis, which is the most commonly used summary evaluation method in the literature.

In addition, the models were evaluated by changing the compression ratio, which corresponds to how much the original documents should be reduced. The features for extractive text summarization include various measures: Term frequency, Title feature, Term Frequency and inverse sentence frequency (TF-ISF), Position frequency, Length feature, Sentence-Sentence similarity, Bushy path, Phrasal information and Sentence count in the document. The feature vector includes: Dataset, preprocessing, Feature selection, labeling, sampling and sentence extraction. The results showed that the neural approach could not achieve a convincing summarization performance as it incorporates extractive text summarization method.

2.3 Abstractive Text Summarization using Sentimental Infusion [10]

The proposed approach is able to leverage the sentence word order to form coherent sentences, and hence the summaries are concise and helps the model to communicate the information in conjunction with the ability to remove the redundancy from the input text. No domain knowledge is required for proposed algorithm to work. It is adaptable to different types of content as well. Also, because it does not use any semantic information related to the sentences of the

input document, it is not able to combine sentences that are semantically related but not related syntactically. To address this shortcoming, abstractive summarization system based semantic representations, like parse trees, Abstract Meaning Representation (AMR) graphs can be used. The methodology consists of building the word graph, Ensuring the sentence correctness and getting the abstractive summaries.

The Methodology consists of three steps namely, first step is Building the word graph, second step is Ensuring the sentence correctness, third is Getting the abstractive summaries. The third process includes calculating the scores of the paths, Fusing Sentiments and Summarization. Once all the paths are scored as well as the sentences have been fused, it ranks the sentences in descending order of their scores and remove duplicate sentences from the summary using Jaccard index for similarity measure. Then the remaining top most S (number of sentences in the summary specified by the user) sentences are chosen for the summary. But it is not able to combine sentences that are semantically related but not related syntactically. The pre-existing extractive techniques can be used to improve recall and abstractive techniques can then be applied to improve precision of the summary.

2.4 Text summarization using deep learning algorithm [11]

The proposed system generates summary for multi-documents, it is developed as an automatic multi-documents summarization system incorporated with the RBM. It consists of four different features for feature extraction phase. The feature score for sentences is applied to the RBM in which the RBM rules are optimized with the help of deep learning algorithm. The features are processed through different levels of the RBM algorithm and the text summary is generated accordingly. The generated result is tested as per the evaluation matrices. The deep learning algorithm consists of optimal feature generation, Summary generation, Sentence score, result and analysis. The experimentation of the

proposed text summarization algorithm is done by considering three different document sets: the evaluation matrices considered in the proposed system are recall, precision and f-measure. The enhancement needed to the proposed approach can be done by considering different features and by adding more hidden layers to the RBM algorithm.

2.5 Text Summarization using neural Network [12]

The model is proposed with a machine learning approach that uses artificial neural networks to produce summaries of arbitrary lengthy new articles. The neural network is then modified, through feature fusion, to produce a summary of highly ranked sentences of the article. Through feature fusion, the network discovers the importance and unimportance of various features used to determine the summary worthiness of each sentence. The first step involves training a neural network to recognize the type of sentences that should be included in the summary, the second step, feature fusion, prunes the neural network and collapses the hidden layer unit activations into discrete values with frequencies. This step generalizes the important features that must exist in the summary sentences by fusing the features and finding trends in the summary sentences. The third step, sentence selection, uses the modified neural network to filter the text and to select only the highly ranked sentences. This step controls the selection of the summary sentences in terms of their sentences.

The neural network consists of seven input-layer neurons, six hidden-layer neurons, and one output-layer neuron. It uses a conjugate gradient method where the energy function is a combination of error function and a penalty function. The addition of the penalty function drives the associated weights of unnecessary connections to very small values while strengthening the rest of the connections. Therefore, the unnecessary connections and neurons can be pruned without affecting the performance of the network. But the resulted summary provides only 80% accuracy.

CHAPTER 3

PROJECT DESCRIPTION

3.1 PROBLEM DEFINITION

While a massive amount of data circulating in the digital space, there is need to develop deep learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages. The issue is not only related to the volume of the data, but also to the diversity of subjects, qualities, and idioms. It is a complex and exhaustive process to gather and perceive any main information from such a massive amount of resources in sufficient time for humans.

3.2 OBJECTIVE OF THE PROJECT

- The intention is to create a coherent and fluent summary having only the main points outlined in the given context
- To develop a trained model such that it truly understands the document and try to express that understanding possibly in a precise way using new phrases and words
- To produce efficient summaries correspondence to loaded word embeddings

3.3 SYSTEM ARCHITECTURE

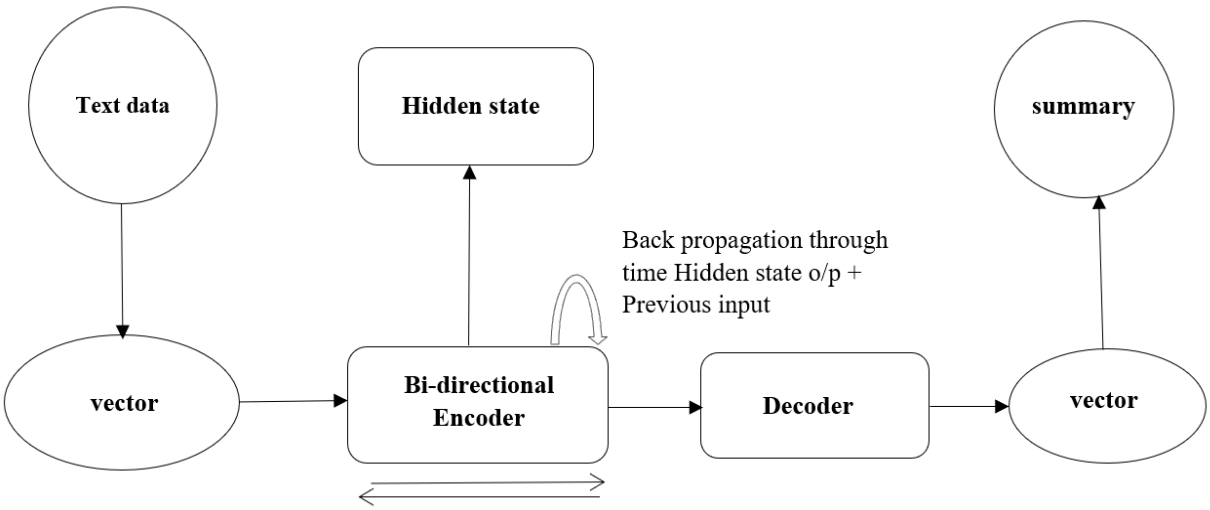


Fig 3.1 System Architecture

3.4 MODULE DESCRIPTION

1. Data Cleaning and Pre-Processing
2. Designing LSTM layer
3. Extracting summary and Evaluation measure

3.4.1 Data Cleaning and Pre-Processing

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Initially the data set consists of 29451 records. The cleaning phase consists of removing stopwords, removing null entries, replacing contractions with their longer forms, and removing unwanted characters like symbols and emojis. Stopwords are only removed from Text part of the dataset, they are not removed from Headlines to sound like more natural phrases. After processing the data, it will have only two columns i.e. text and headlines.

Once the data is cleaned, it is tokenized and converted into vectors so that it can be processed by the model. Then load the pre-trained ConceptNet

Numberbatch word embedding. And find the number of words missing in the embedding by comparing all the tokens from the dataset to the loaded embedding. Also add <unk> token for those which words are not known and add <pad> in the end of line to indicate that it's the end of line. Finally, this will sort the summary and text by the length of text, i.e. from shortest to longest.

In the next phase, the model designed will be used while training. Since the project is built using TensorFlow, the major effort goes into building graphs. Create few place holders to hold data and other parameters. Then create the bi-directional encoder, which will generate its own representation of input data.

3.4.2 Designing LSTM layer

The designing phase deals with the modelling of Attentive Recurrent architecture consisting of recurrent decoder and attentive encoder. While designing the decoder the Bahdanau Attention is implemented. Typically the conditional probability is modelled by a parametric function with parameters θ: $P(y|x) = P(y|x;\theta)$, where y is the target sequence and x is the input sentence sequence. Training involves finding the θ which maximizes the conditional probability of sentence-summary pairs in the training corpus. If the model is trained to generate the next word of the summary, given the previous words, then the above conditional can be factorized into a product of individual conditional probabilities. This conditional probability is implemented using RNN Encoder-Decoder. It is also called as Recurrent Attentive summarizer. The designed model is trained using the training dataset with the graph built and loss graph is generated based on the loss in training data. In the next phase, the generated model is tested using the testing data and the extracted summary is evaluated using the ROUGE score.

3.4.3 Extracting summary and Evaluation measure

The details of the encoder which computes the context vector for every time step is given to the decoder. These vectors can be the representation of the word which captures the position in which it occurs in the sentence and the context in which it appears in the sentence. The vectors are aggregated to form a context vector which helps in the extraction of summary from the designed model. The extracted summary replicates the actual data in short form using the context vector generated. However, this type of model has two shortcomings like producing factual details inaccurately and they tend to repeat themselves. These shortcomings can be corrected by rigorous and continuous training. The generated summary is the gist of the actual text part developed by the model to provide the user a better understanding about the long passages of news.

Evaluation measure is a very important task in the field of automatic summarization of text. Evaluating the summary besides enhancing development of reusable resources and infrastructure helps in comparing and replicating results and thus, adds competition to improve the results. However, it is practically impossible to manually evaluate multiple documents for obtaining an unbiased view. Therefore, reliable automatic evaluation metrics are required for fast and consistent evaluation. To evaluate the project results, the ROUGE score (standard metric) is used to generate scores with respect to the results produced by the model.

CHAPTER 4

PERFORMANCE EVALUATION

4.1 EXISTING SYSTEM

The proliferation of the Internet has become increasingly difficult to efficiently reach information from such massive amounts of data. It is a complex and exhaustive process to gather and perceive any main information from such a massive amount of resources in sufficient time for humans. The rising quantity of information causes some performance issues, such as insufficient solutions and unwieldy applications of information retrieval tasks. It is highly crucial that the resulting document has high cohesion, to be readable and understandable for the reader. The existing methods uses LSTM algorithm with Word2vec and GloVe word embeddings to train the models.

4.2 PROPOSED SYSTEM

The proposed system is built on the concept of Deep Learning for abstractive summarizer. The concept of deep learning trains the machines with some data which makes it capable of producing recapitulation of text document. Neural sequence-to-sequence models have provided a viable approach for abstractive text summarization. This project work uses a Long Short-Term Memory (LSTM) sequence-to-sequence attention model. The method utilizes a local attention model for generating each word of the summary conditioned on the input sentence. While the model is structurally simple, it can easily be trained end-to-end and scales to a large amount of training data. The reconstructed paragraph is evaluated using standard metrics like ROUGE, showing that neural models can encode texts in a way that preserve syntactic, semantic, and discourse coherence.

4.3 METHODOLOGY

Neural Networks (NN) are also used for Natural Language Processing (NLP), including Summarizers. Neural networks are effective in solving almost any machine learning classification problem. Important parameters required in defining the architecture of neural network (NN) are number of hidden layers to be used, number of hidden units to be present in each layer, activation function for each node, error threshold for the data, the type of interconnections, etc. neural networks can capture very complex characteristics of data without any significant involvement of manual labor as opposed to the machine learning systems. Deep learning uses deep neural networks to learn good representations of the input data, which can then be used to perform specific tasks.

4.3.1 Recurrent Neural Network (RNN)

Recurrent Neural Networks were created in the 1980's but have just been recently gaining popularity from advances to the networks designs and increased computational power from graphic processing units. They are especially useful with sequential data because each neuron or unit can use its internal memory to maintain information about the previous input. This allows the network to gain a deeper understanding. This is important to note because reading through a sentence even as a human, picking up the context of each word from the words before it. An RNN has loops in them that allow information to be carried across neurons while reading in input. Theoretically RNNs can handle context from the beginning of the sentence which will allow more accurate predictions of a word at the end of a sentence. In practice this isn't necessarily true for vanilla RNNs. This is a major reason why RNNs faded out from practice for a while until some great results were achieved with using a Long Short-Term Memory (LSTM) unit inside the Neural Network.[16]

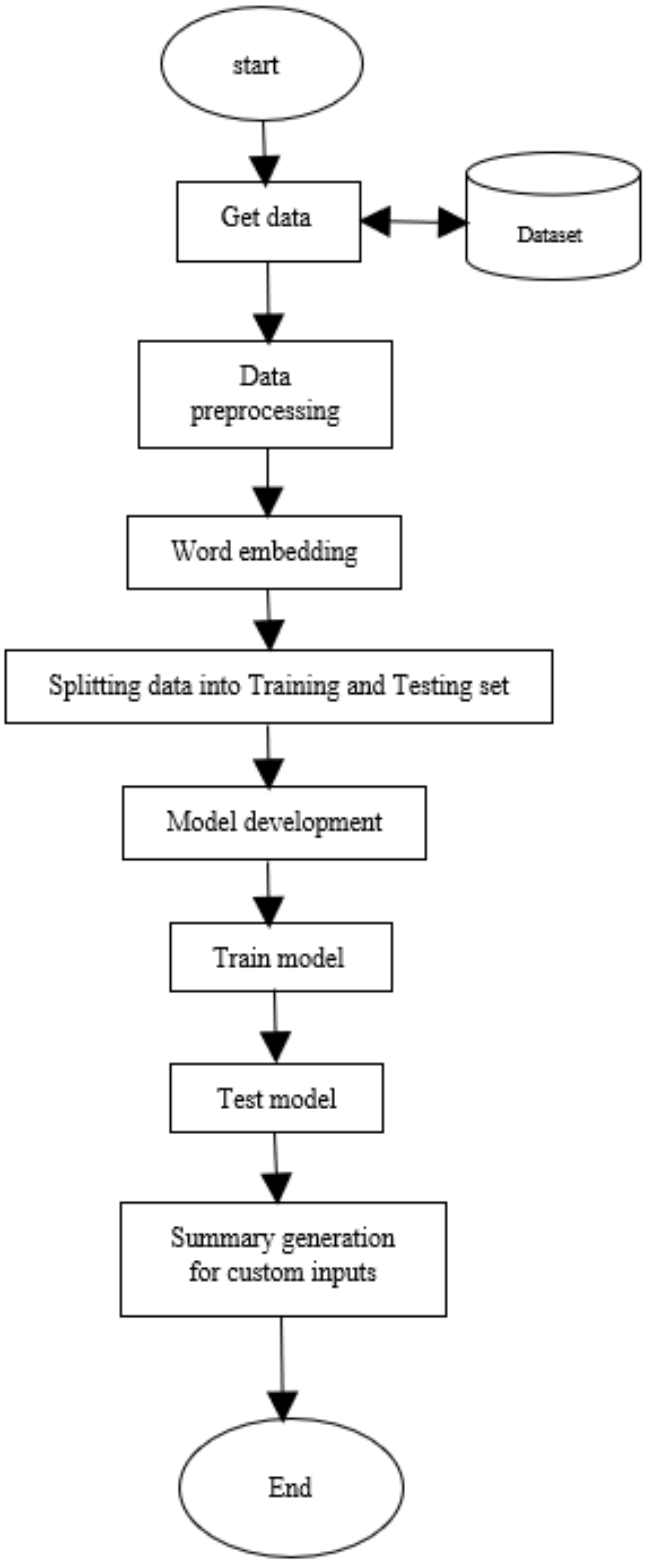


Fig. 4.1 Work Flow

4.3.2 Long Short-Term Memory (LSTM) Units

The LSTM is RNN architecture which can remember past contextual values. These stored values do not change over time while training the model. There are four components in LSTM which are LSTM Units, LSTM Blocks, LSTM Gates and LSTM Recurrent Components. LSTM Unit store values for long time or for short time. LSTM has no activation functions for their recurrent components. Since there are no activation function the values of units does not change for some period until the context is changed. A LSTM Block contains such many units. LSTM's are considered as deep neural networks. These LSTM's are implemented in parallel systems. LSTM blocks have four gates to control the information flow.

Logistic functions are used to implement these gates, to compute a value between 0 and 1. To allow or deny information flow into or out of the memory, multiplication of values with these logistic functions are done. To control the flow of new values into memory, input gate plays key role. The extent to which a value remains in memory is controlled by forget gate. Output gate controls the extent to which the value in memory is used to compute the output activation of the block. Sometimes in implementations, the input and forget gates are merged into a single gate, hence we can see even 3 gate representations of LSTM. When new value which is worth remembering is available then we can forget the old value. This represents the combining effect of input and forget gate of LSTM.

4.3.3 Encoders and Decoders

One approach to seq2seq prediction problems that has proven very effective is called the Encoder-Decoder LSTM. This architecture is comprised of two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. The use of the models in concert gives the architecture its

name of Encoder-Decoder LSTM designed specifically for seq2seq problems. The Encoder-Decoder LSTM was developed for natural language processing problems where it demonstrated state-of-the-art performance, specifically in text translation called statistical machine translation. The innovation of this architecture is the use of a fixed-sized internal representation in the heart of the model that input sequences are read to and output sequences are read from. For this reason, the method may be referred to as sequence embedding.

In one of the first applications of the architecture to English-to-French translation, the internal representation of the encoded English phrases was visualized. The plots revealed a qualitatively meaningful learned structure of the phrases harnessed for the translation task. On the task of translation, the model was found to be more effective when the input sequence was reversed. Further, the model was shown to be effective even on very long input sequences. This approach has also been used with image inputs where a Convolutional Neural Network is used as a feature extractor on input images, which is then read by a decoder LSTM. Bidirectional recurrent neural networks (RNN) are just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation. This structure allows the networks to have both backward and forward information about the sequence at every time step.[17]

4.4.4 Word Embeddings

The abstractive text summarization project uses ConceptNet Numberbatch word embedding. ConceptNet Numberbatch is part of the ConceptNet open data project. ConceptNet provides lots of ways to compute with word meanings, one of which is word embeddings. ConceptNet Numberbatch is a snapshot of just the word embeddings. It is built using an ensemble that combines data from

ConceptNet, word2vec, GloVe, and OpenSubtitles 2016, using a variation on retrofitting.

4.5 EXPERIMENTAL SETUP

- **Python**

Python is a high-level, interpreted programming language, created by Guido van Rossum. The language is very popular for its code readability and compact line of codes. It uses white space inundation to delimit blocks. Python provides a large standard library which can be used for various applications for example natural language processing, machine learning, data analysis etc. It is favoured for complex projects, because of its simplicity, diverse range of features and its dynamic nature. The project uses Google Colaboratory to develop the model.

- **Google Colaboratory**

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. Colab notebooks are stored in Google Drive, or can be loaded from GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook, or follow these Google Drive file sharing instructions.

For abstractive text summarization, the ConceptNet Numberbatch word embedding similar to GloVe is used. The generated summary is evaluated using the ROUGE score which provides a value of 0.5106. The precision and recall values are 0.4 and 0.705 which shows that the developed model provides a better result with the developed algorithm (LSTM). The developed model on comparison with existing models provides a better result. The improved model performance with the existing methods is shown as a comparison graph below:

Fig. 4.2 Performance Evaluation chart

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

The system aims at implementing a state-of-the-art model for abstractive sentence summarization to a recurrent neural network architecture. The model is a simplified version of the encoder-decoder framework for machine translation. The model is trained on the News_summary corpus to generate summaries of news based on the contents in the text part. Evaluation measures are implemented to enhance the extractive text summarization by improving: summary representation, sentence ranking, and sentence selection process. However, Abstractive text summarization is a challenging area because of the complexity of natural language processing. The developed model can be used in real time applications like Food reviews, Headline generation, etc.

After reviewing the results produced by abstractive text summarizer, it can be concluded that sometimes repetition of words occur when they deal with large text documents. To handle such large documents more research work is needed by using advanced deep learning techniques. There is a need to propose new methods while considering minute details in massive. To achieve this, a hybrid model that is both able to handle enormous dataset and the real time summary generation problem can be included to have a response during large data processing with an improved accuracy.

APPENDIX 1

SOFTWARE AND HARDWARE SPECIFICATION

Software Specification

- Google Colaboratory

Hardware Specification

- Processor: Intel(R)Core(TM)i5
- RAM 4GB

Dataset

Name: News_summary

Link: https://www.kaggle.com/sunnysai12345/news-summary

Dataset description

The news dataset has been taken from kaggle.com. The dataset is in the format of .csv and it is highly skewed. It contains 29451 records with 6 variables such as Author, Date, Headlines, Read_more, Text, cText.

APPENDIX 2

SOURCE CODE

Importing packages:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.python.layers.core import Dense
from tensorflow.python.ops.rnn_cell_impl import _zero_state_tensors
import re
from nltk.corpus import stopwords
import time
print('TensorFlow Version: {}'.format(tf.__version__))
import csv
```

Data cleaning:

```
def clean_text(text, remove_stopwords):
    # Convert words to lower case
  text = text.lower()
    # Replace contractions with their longer forms
  if True:
    text = text.split()
    new_text = []
    for word in text:
      if word in contractions:
        new_text.append(contractions[word])
      else:
        new_text.append(word)
    text = " ".join(new_text)
    # Format words and remove unwanted characters
  text = re.sub(r'https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE)
```

```
    text = re.sub(r'\<a href', ' ', text)
    text = re.sub(r'&amp;', '', text)
    text = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\]/]', ' ', text)
    text = re.sub(r'<br />', ' ', text)
    text = re.sub(r'\', ' ', text)
        # Optionally, remove stop words
    if remove_stopwords:
        text = text.split()
        stops = set(stopwords.words("english"))
        text = [w for w in text if not w in stops]
        text = " ".join(text)
    return text
clean_summaries = []
for headlines in mail.headlines:
    clean_summaries.append(clean_text(str(headlines), remove_stopwords=False
))
print("Headlines are complete.")
clean_texts = []
for text in mail.text:
    clean_texts.append(clean_text(str(text), remove_stopwords=True))
print("Texts are complete.")
```

Word Embedding:

```
embeddings_index = {}
with open('drive/My Drive/Colab Notebooks/numberbatch-
en.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split(' ')
        word = values[0]
        embedding = np.asarray(values[1:], dtype='float32')
```

```
        embeddings_index[word] = embedding
print('Word embeddings:', len(embeddings_index))
```

Finding number of words missing from CN:

```
missing_words = 0
threshold = 20
for word, count in word_counts.items():
    if count > threshold:
        if word not in embeddings_index:
            missing_words += 1
missing_ratio = round(missing_words/len(word_counts),4)*100
print("Number of words missing from CN:", missing_words)
print("Percent of words that are missing from vocabulary: {}%".format(missing
_ratio))
```

Building the Model:

```
def model_inputs():
    '''Create palceholders for inputs to the model'''
    input_data = tf.placeholder(tf.int32, [None, None], name='input')
    targets = tf.placeholder(tf.int32, [None, None], name='targets')
    lr = tf.placeholder(tf.float32, name='learning_rate')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')
    summary_length = tf.placeholder(tf.int32, (None,), name='summary_length')
    max_summary_length = tf.reduce_max(summary_length, name='max_dec_le
n')
    text_length = tf.placeholder(tf.int32, (None,), name='text_length')

    return input_data, targets, lr, keep_prob, summary_length, max_summary_le
ngth, text_length

def process_encoding_input(target_data, vocab_to_int, batch_size):
```

```
    ending = tf.strided_slice(target_data, [0, 0], [batch_size, -1], [1, 1])
    dec_input = tf.concat([tf.fill([batch_size, 1], vocab_to_int['<GO>']), ending],
1)
    return dec_input
```

Create encoding layer:

```
def encoding_layer(rnn_size, sequence_length, num_layers, rnn_inputs, keep_p
rob):
    for layer in range(num_layers):
        with tf.variable_scope('encoder_{}'.format(layer)):
            cell_fw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-
0.1, 0.1, seed=2))
            cell_fw = tf.contrib.rnn.DropoutWrapper(cell_fw,
                                                    input_keep_prob = keep_prob)

            cell_bw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-
0.1, 0.1, seed=2))
            cell_bw = tf.contrib.rnn.DropoutWrapper(cell_bw,
                                                    input_keep_prob = keep_prob)

            enc_output, enc_state = tf.nn.bidirectional_dynamic_rnn(cell_fw,
                                                            cell_bw,
                                                            rnn_inputs,
                                                            sequence_length,
                                                            dtype=tf.float32)
    # Join outputs since we are using a bidirectional RNN
    enc_output = tf.concat(enc_output,2)
        return enc_output, enc_state
```

Create decoding layer:

```
def training_decoding_layer(dec_embed_input, summary_length, dec_cell, initi
al_state, output_layer,
                            vocab_size, max_summary_length):
    training_helper = tf.contrib.seq2seq.TrainingHelper(inputs=dec_embed_in
put, sequence_length=summary_length,time_major=False)

    training_decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell,
                                                       training_helper,
                                                       initial_state,
                                                       output_layer)
    training_logits, _ , _ = tf.contrib.seq2seq.dynamic_decode(training_decoder,
                                                           output_time_major=False,
                                                           impute_finished=True,
                                                           maximum_iterations=max_summary_lengt
h)
    return training_decoder

def inference_decoding_layer(embeddings, start_token, end_token, dec_cell, ini
tial_state, output_layer,
                             max_summary_length, batch_size):

    start_tokens = tf.tile(tf.constant([start_token], dtype=tf.int32), [batch_size], n
ame='start_tokens')

    inference_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embeddings,
                                                            start_tokens,
                                                            end_token)

    inference_decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell,
                                                        inference_helper,
                                                        initial_state,
                                                        output_layer)
```

```
    inference_logits, _ , _ = tf.contrib.seq2seq.dynamic_decode(inference_decod
er,
                                                        output_time_major=False,
                                                        impute_finished=True,
                                                        maximum_iterations=max_summary_leng
th)

    return inference_decoder

#output_units=12
def decoding_layer(dec_embed_input, embeddings, enc_output, enc_state, voca
b_size, text_length, summary_length,
                   max_summary_length, rnn_size, vocab_to_int, keep_prob, batch_si
ze, num_layers):
    '''Create the decoding cell and attention for the training and inference decodin
g layers'''

    for layer in range(num_layers):
        with tf.variable_scope('decoder_{}'.format(layer)):
            lstm = tf.contrib.rnn.LSTMCell(rnn_size,
                                           initializer=tf.random_uniform_initializer(-
0.1, 0.1, seed=2))
            dec_cell = tf.contrib.rnn.DropoutWrapper(lstm,
                                                     input_keep_prob = keep_prob)

    output_layer = Dense(vocab_size,
                         kernel_initializer = tf.truncated_normal_initializer(mean = 0.0, s
tddev=0.1))

    attn_mech = tf.contrib.seq2seq.BahdanauAttention(rnn_size,
                                                     enc_output,
                                                     text_length,
```

```
                                    normalize=False,
                                    name='BahdanauAttention')

    dec_cell = tf.contrib.seq2seq.AttentionWrapper(dec_cell,
                                           attn_mech,
                                           rnn_size)
    initial_state = dec_cell.zero_state(batch_size=batch_size,dtype=tf.float32).clo
ne(cell_state=enc_state[0])

    with tf.variable_scope("decode"):
        training_decoder = training_decoding_layer(dec_embed_input,
                                    summary_length,
                                    dec_cell,
                                    initial_state,
                                    output_layer,
                                    vocab_size,
                                    max_summary_length)

        training_logits,_ ,_ = tf.contrib.seq2seq.dynamic_decode(training_decoder,
                            output_time_major=False,
                            impute_finished=True,
                            maximum_iterations=max_summary_length)
    with tf.variable_scope("decode", reuse=True):
        inference_decoder = inference_decoding_layer(embeddings,
                                     vocab_to_int['<GO>'],
                                     vocab_to_int['<EOS>'],
                                     dec_cell,
                                     initial_state,
                                     output_layer,
```

```
                                    max_summary_length,
                                    batch_size)

    inference_logits,_ ,_ = tf.contrib.seq2seq.dynamic_decode(inference_deco
der,
                                output_time_major=False,
                                impute_finished=True,
                                maximum_iterations=max_summary_length)

  return training_logits, inference_logits
```

Sequence-to-sequence model:

```
def seq2seq_model(input_data, target_data, keep_prob, text_length, summary_l
ength, max_summary_length,
                  vocab_size, rnn_size, num_layers, vocab_to_int, batch_size):
  '''Use the previous functions to create the training and inference logits'''

  # Use Numberbatch's embeddings and the newly created ones as our embeddi
ngs
  embeddings = word_embedding_matrix

  enc_embed_input = tf.nn.embedding_lookup(embeddings, input_data)
  enc_output, enc_state = encoding_layer(rnn_size, text_length, num_layers, en
c_embed_input, keep_prob)

  dec_input = process_encoding_input(target_data, vocab_to_int, batch_size)
  dec_embed_input = tf.nn.embedding_lookup(embeddings, dec_input)

    training_logits, inference_logits  = decoding_layer(dec_embed_input,
                                        embeddings,
```

```
                                    enc_output,
                                    enc_state,
                                    vocab_size,
                                    text_length,
                                    summary_length,
                                    max_summary_length,
                                    rnn_size,
                                    vocab_to_int,
                                    keep_prob,
                                    batch_size,
                                    num_layers)

    return training_logits, inference_logits
```

Graph Building:

```
  # Load the model inputs
    input_data, targets, lr, keep_prob, summary_length, max_summary_length, te
xt_length = model_inputs()

    # Create the training and inference logits
    training_logits, inference_logits = seq2seq_model(tf.reverse(input_data, [-1]),
                                      targets,
                                      keep_prob,
                                      text_length,
                                      summary_length,
                                      max_summary_length,
                                      len(vocab_to_int)+1,
                                      rnn_size,
                                      num_layers,
                                      vocab_to_int,
```

```
                                                batch_size)

    # Create tensors for the training logits and inference logits
    training_logits = tf.identity(training_logits.rnn_output, 'logits')
    inference_logits = tf.identity(inference_logits.sample_id, name='predictions')

    # Create the weights for sequence_loss
    masks = tf.sequence_mask(summary_length, max_summary_length, dtype=tf
.float32, name='masks')

    with tf.name_scope("optimization"):
        # Loss function
        cost = tf.contrib.seq2seq.sequence_loss(
            training_logits,
            targets,
            masks)

        # Optimizer
        optimizer = tf.train.AdamOptimizer(learning_rate)

        # Gradient Clipping
        gradients = optimizer.compute_gradients(cost)
        capped_gradients = [(tf.clip_by_value(grad, -
5., 5.), var) for grad, var in gradients if grad is not None]
        train_op = optimizer.apply_gradients(capped_gradients)
print("Graph is built.")
```

Training the graph:

```
# Train the Model
learning_rate_decay = 0.95
```

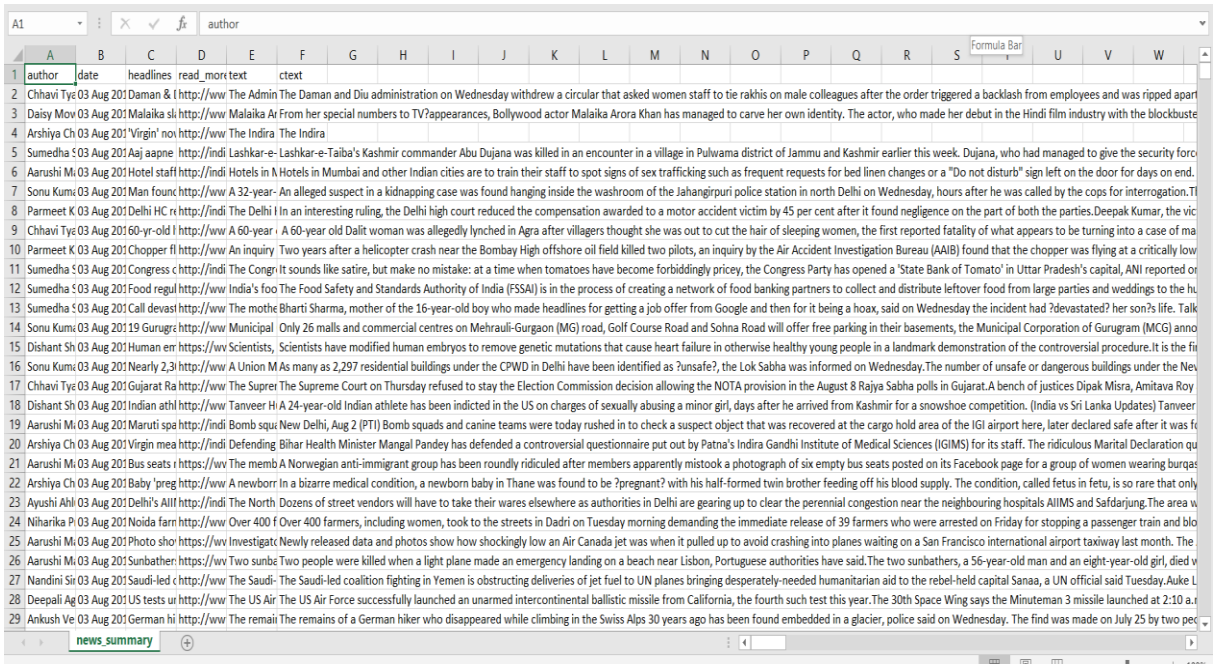
```
min_learning_rate = 0.0005
display_step = 5 # Check training loss after every 20 batches
stop_early = 0
stop = 3 # If the update loss does not decrease in 3 consecutive update checks, st
op training
per_epoch = 3 # Make 3 update checks per epoch
update_check = (len(sorted_texts)//batch_size//per_epoch)-1
update_loss = 0
batch_loss = 0
summary_update_loss = [] # Record the update losses for saving improvements
in the model
checkpoint = "./best_model.ckpt"
print("Training will Strat now.")
with tf.Session(graph=train_graph) as sess:
    sess.run(tf.global_variables_initializer())
    for epoch_i in range(1, epochs+1):
        update_loss = 0
        batch_loss = 0
        for batch_i, (summaries_batch, texts_batch, summaries_lengths, texts_leng
ths) in enumerate(
                get_batches(sorted_summaries, sorted_texts, batch_size)):
            start_time = time.time()
            _, loss = sess.run(
                [train_op, cost],
                {input_data: texts_batch,
                 targets: summaries_batch,
                 lr: learning_rate,
                 summary_length: summaries_lengths,
                 text_length: texts_lengths,
```

```
            keep_prob: keep_probability})
        batch_loss += loss
        update_loss += loss
        end_time = time.time()
        batch_time = end_time - start_time

        if batch_i % display_step == 0 and batch_i > 0:
            print('Epoch {:>3}/{} Batch {:>4}/{} - Loss: {:>6.3f}, Seconds: {:>4.
2f}'
                  .format(epoch_i,
                          epochs,
                          batch_i,
                          len(sorted_texts) // batch_size,
                          batch_loss / display_step,
                          batch_time*display_step))
            batch_loss = 0

        if batch_i % update_check == 0 and batch_i > 0:
            print("Average loss for this update:", round(update_loss/update_check
,3))
            summary_update_loss.append(update_loss)

            # If the update loss is at a new minimum, save the model
            if update_loss <= min(summary_update_loss):
                print('New Record!')
                stop_early = 0
                saver = tf.train.Saver()
                saver.save(sess, checkpoint)
```

```
            else:
                print("No Improvement.")
                stop_early += 1
                if stop_early == stop:
                    break
            update_loss = 0
        saver = tf.train.Saver()
        saver.save(sess, checkpoint)
        # Reduce learning rate, but not below its minimum value
        learning_rate *= learning_rate_decay
        if learning_rate < min_learning_rate:
            learning_rate = min_learning_rate
        if stop_early == stop:
            print("Stopping Training.")
            break
print("Model Trained")
```

APPENDIX 3

SCREENSHOTS



	author	date	headlines	read_more	text	ctext
2	Chhavi Tya	03 Aug 201	Daman & I	http://ww	The Admin	The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after the order triggered a backlash from employees and was ripped apart
3	Daisy Mov	03 Aug 201	Malaika sh	http://ww	Malaika Ar	From her special numbers to TV?appearances, Bollywood actor Malaika Arora Khan has managed to carve her own identity. The actor, who made her debut in the Hindi film industry with the blockbuste
4	Arshiya Ch	03 Aug 201	'Virgin' no	http://ww	The Indira	The Indira
5	Sumedha S	03 Aug 201	Aaj aapne	http://indi	Lashkar-e-	Lashkar-e-Taiba's Kashmir commander Abu Dujana was killed in an encounter in a village in Pulwama district of Jammu and Kashmir earlier this week. Dujana, who had managed to give the security forc
6	Aarushi Ma	03 Aug 201	Hotel staff	http://indi	Hotels in M	Hotels in Mumbai and other Indian cities are to train their staff to spot signs of sex trafficking such as frequent requests for bed linen changes or a "Do not disturb" sign left on the door for days on end.
7	Sonu Kuma	03 Aug 201	Man found	http://ww	A 32-year-	An alleged suspect in a kidnapping case was found hanging inside the washroom of the Jahangirpuri police station in north Delhi on Wednesday, hours after he was called by the cops for interrogation.T
8	Parmeet K	03 Aug 201	Delhi HC re	http://indi	The Delhi I	In an interesting ruling, the Delhi high court reduced the compensation awarded to a motor accident victim by 45 per cent after it found negligence on the part of both the parties.Deepak Kumar, the vic
9	Chhavi Tya	03 Aug 201	60-yr-old I	http://ww	A 60-year	A 60-year old Dalit woman was allegedly lynched in Agra after villagers thought she was out to cut the hair of sleeping women, the first reported fatality of what appears to be turning into a case of ma
10	Parmeet K	03 Aug 201	Chopper fl	http://ww	An inquiry	Two years after a helicopter crash near the Bombay High offshore oil field killed two pilots, an inquiry by the Air Accident Investigation Bureau (AAIB) found that the chopper was flying at a critically low
11	Sumedha S	03 Aug 201	Congress c	http://indi	The Congr	It sounds like satire, but make no mistake: at a time when tomatoes have become forbiddingly pricey, the Congress Party has opened a 'State Bank of Tomato' in Uttar Pradesh's capital, ANI reported or
12	Sumedha S	03 Aug 201	Food regul	http://ww	India's foo	The Food Safety and Standards Authority of India (FSSAI) is in the process of creating a network of food banking partners to collect and distribute leftover food from large parties and weddings to the hu
13	Sumedha S	03 Aug 201	Call devast	http://ww	The mothe	Bharti Sharma, mother of the 16-year-old boy who made headlines for getting a job offer from Google and then for it being a hoax, said on Wednesday the incident had ?devastated? her son?s life. Talk
14	Sonu Kuma	03 Aug 201	19 Gurugra	http://ww	Municipal	Only 26 malls and commercial centres on Mehrauli-Gurgaon (MG) road, Golf Course Road and Sohna Road will offer free parking in their basements, the Municipal Corporation of Gurugram (MCG) anno
15	Dishant Sh	03 Aug 201	Human em	https://ww	Scientists,	Scientists have modified human embryos to remove genetic mutations that cause heart failure in otherwise healthy young people in a landmark demonstration of the controversial procedure.It is the fi
16	Sonu Kuma	03 Aug 201	Nearly 2,3	http://ww	A Union M	As many as 2,297 residential buildings under the CPWD in Delhi have been identified as ?unsafe?, the Lok Sabha was informed on Wednesday.The number of unsafe or dangerous buildings under the Ne
17	Chhavi Tya	03 Aug 201	Gujarat Ra	http://ww	The Supren	The Supreme Court on Thursday refused to stay the Election Commission decision allowing the NOTA provision in the August 8 Rajya Sabha polls in Gujarat.A bench of justices Dipak Misra, Amitava Roy
18	Dishant Sh	03 Aug 201	Indian athl	http://ww	Tanveer Hi	A 24-year-old Indian athlete has been indicted in the US on charges of sexually abusing a minor girl, days after he arrived from Kashmir for a snowshoe competition. (India vs Sri Lanka Updates) Tanveer
19	Aarushi Ma	03 Aug 201	Maruti spa	http://indi	Bomb squa	New Delhi, Aug 2 (PTI) Bomb squads and canine teams were today rushed in to check a suspect object that was recovered at the cargo hold area of the IGI airport here, later declared safe after it was fe
20	Arshiya Ch	03 Aug 201	Virgin mea	http://indi	Defending	Bihar Health Minister Mangal Pandey has defended a controversial questionnaire put out by Patna's Indira Gandhi Institute of Medical Sciences (IGIMS) for its staff. The ridiculous Marital Declaration qu
21	Aarushi Ma	03 Aug 201	Bus seats r	https://ww	The memb	A Norwegian anti-immigrant group has been roundly ridiculed after members apparently mistook a photograph of six empty bus seats posted on its Facebook page for a group of women wearing burqas
22	Arshiya Ch	03 Aug 201	Baby 'preg	http://ww	A newborn	In a bizarre medical condition, a newborn baby in Thane was found to be ?pregnant? with his half-formed twin brother feeding off his blood supply. The condition, called fetus in fetu, is so rare that only
23	Ayushi Ahl	03 Aug 201	Delhi's AIII	http://indi	The North	Dozens of street vendors will have to take their wares elsewhere as authorities in Delhi are gearing up to clear the perennial congestion near the neighbouring hospitals AIIMS and Safdarjung.The area w
24	Niharika P	03 Aug 201	Noida farn	http://ww	Over 400 f	Over 400 farmers, including women, took to the streets in Dadri on Tuesday morning demanding the immediate release of 39 farmers who were arrested on Friday for stopping a passenger train and blo
25	Aarushi Ma	03 Aug 201	Photo sho	https://ww	Investigate	Newly released data and photos show how shockingly low an Air Canada jet was when it pulled up to avoid crashing into planes waiting on a San Francisco international airport taxiway last month. The
26	Aarushi Ma	03 Aug 201	Sunbather	https://ww	Two sunbe	Two people were killed when a light plane made an emergency landing on a beach near Lisbon, Portuguese authorities have said.The two sunbathers, a 56-year-old man and an eight-year-old girl, died w
27	Nandini Sir	03 Aug 201	Saudi-led c	http://ww	The Saudi-	The Saudi-led coalition fighting in Yemen is obstructing deliveries of jet fuel to UN planes bringing desperately-needed humanitarian aid to the rebel-held capital Sanaa, a UN official said Tuesday.Auke L
28	Deepali Ag	03 Aug 201	US tests ur	http://ww	The US Air	The US Air Force successfully launched an unarmed intercontinental ballistic missile from California, the fourth such test this year.The 30th Space Wing says the Minuteman 3 missile launched at 2:10 a.r
29	Ankush Ve	03 Aug 201	German hi	http://ww	The remai	The remains of a German hiker who disappeared while climbing in the Swiss Alps 30 years ago has been found embedded in a glacier, police said on Wednesday. The find was made on July 25 by two pe

Fig A3.1 Original Dataset

	headlines	text
0	upGrad learner switches to career in ML & Al w...	Saurav Kant, an alumnus of upGrad and IIIT-B's...
1	Delhi techie wins free food from Swiggy for on...	Kunal Shah's credit card bill payment platform...
2	New Zealand end Rohit Sharma-led India's 12-ma...	New Zealand defeated India by 8 wickets in the...
3	Aegon life iTerm insurance plan helps customer...	With Aegon Life iTerm Insurance plan, customer...
4	Have known Hirani for yrs, what if MeToo claim...	Speaking about the sexual harassment allegatio...

Fig A3.2 Cleaned Data

```
[ ] Epoch  25/25 Batch  140/276 - Loss:  1.526, Seconds: 0.67
    Epoch  25/25 Batch  145/276 - Loss:  1.618, Seconds: 0.69
    Epoch  25/25 Batch  150/276 - Loss:  1.717, Seconds: 0.68
    Epoch  25/25 Batch  155/276 - Loss:  1.657, Seconds: 0.68
    Epoch  25/25 Batch  160/276 - Loss:  1.645, Seconds: 0.69
    Epoch  25/25 Batch  165/276 - Loss:  1.616, Seconds: 0.70
    Epoch  25/25 Batch  170/276 - Loss:  1.597, Seconds: 0.69
    Epoch  25/25 Batch  175/276 - Loss:  1.622, Seconds: 0.70
    Epoch  25/25 Batch  180/276 - Loss:  1.651, Seconds: 0.70
    Average loss for this update: 1.634
    New Record!
    Epoch  25/25 Batch  185/276 - Loss:  1.637, Seconds: 0.73
    Epoch  25/25 Batch  190/276 - Loss:  1.634, Seconds: 0.69
    Epoch  25/25 Batch  195/276 - Loss:  1.639, Seconds: 0.72
    Epoch  25/25 Batch  200/276 - Loss:  1.661, Seconds: 0.71
    Epoch  25/25 Batch  205/276 - Loss:  1.683, Seconds: 0.71
    Epoch  25/25 Batch  210/276 - Loss:  1.640, Seconds: 0.71
    Epoch  25/25 Batch  215/276 - Loss:  1.619, Seconds: 0.79
    Epoch  25/25 Batch  220/276 - Loss:  1.644, Seconds: 0.72
    Epoch  25/25 Batch  225/276 - Loss:  1.630, Seconds: 0.73
    Epoch  25/25 Batch  230/276 - Loss:  1.560, Seconds: 0.76
    Epoch  25/25 Batch  235/276 - Loss:  1.671, Seconds: 0.80
    Epoch  25/25 Batch  240/276 - Loss:  1.596, Seconds: 0.75
    Epoch  25/25 Batch  245/276 - Loss:  1.563, Seconds: 0.75
    Epoch  25/25 Batch  250/276 - Loss:  1.604, Seconds: 0.77
    Epoch  25/25 Batch  255/276 - Loss:  1.639, Seconds: 0.82
    Epoch  25/25 Batch  260/276 - Loss:  1.582, Seconds: 0.76
    Epoch  25/25 Batch  265/276 - Loss:  1.679, Seconds: 0.79
    Epoch  25/25 Batch  270/276 - Loss:  1.629, Seconds: 0.85
    Average loss for this update: 1.627
    New Record!
    Epoch  25/25 Batch  275/276 - Loss:  1.694, Seconds: 0.92
    Model Trained
```

Fig A3.3 Training snippet

```
[ ] import matplotlib.pyplot as plt
    plt.plot(summary_update_loss[:])
    plt.ylabel('Loss')
    plt.show()
```

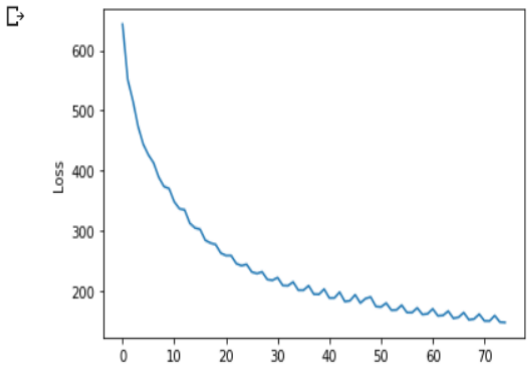


Fig A3.4 Loss at every iteration

```
    print('  Response Words: {} '.format('  '.join([int_to_vocab[i] for i in answer_logits if i != pad])))
    print('\n\n')
    count = count + 1
```

```
INFO:tensorflow:Restoring parameters from ./best_model.ckpt
Original Text: Priyanka Chopra and fiancÃ© Nick Jonas have finalised Umaid Bhawan in Jodhpur as the venue for their destination wedding, as per
Original summary: Priyanka, Nick to tie knot at Jodhpur's Umaid Bhawan: Report

Text
  Word Ids:    [1946, 2648, 8452, 1999, 2000, 7680, 10861, 10795, 7081, 1795, 9852, 872, 2274, 1005, 605, 13904, 2816, 4526, 2004, 240, 230, 2
  Input Words: priyanka chopra fiancã© nick jonas finalised umaid bhawan jodhpur venue destination wedding per reports couple reportedly limite

Summary
  Word Ids:       [1946, 1999, 3, 1651, 8703]
  Response Words: priyanka nick to tie knot



INFO:tensorflow:Restoring parameters from ./best_model.ckpt
Original Text: The Central Government on Thursday stated that over 60,000 people were successfully rescued and shifted to relief camps in the '
Original summary: Saved 60,000 people during rescue ops in Kerala: Centre

Text
  Word Ids:    [6449, 1078, 12441, 17681, 3956, 628, 230, 3288, 930, 1879, 1555, 13306, 9283, 1336, 1555, 1337, 2054, 196, 2548, 838, 776, 1978
  Input Words: central government thursday stated 60 000 people successfully rescued shifted relief camps massive rescue relief operations laun

Summary
  Word Ids:       [196, 3, 1336, 78, 628, 2758, 5]
  Response Words: centre to rescue 1 000 personnel in



INFO:tensorflow:Restoring parameters from ./best_model.ckpt
Original Text: Nitin Sandesara, the promoter of Gujarat-based pharmaceutical company Sterling Biotech which is being probed for â¹5,383 crore
Original summary: Gujarat businessman held in Dubai for â¹5,300 crore fraud
```

Fig A3.5 Summary generated using test data

```
Cleaned Text [113, 626, 1687, 68, 35, 1644, 1519, 1739, 0, 800, 2162, 1519, 1739, 1375, 24, 471]
INFO:tensorflow:Restoring parameters from ./best_model.ckpt
Index Value:  10997

Original Text: I like the slight pineapple flavor in this better than the plain coconut water. Good well chilled. Coconut wat
is supposed to be a healthy drink.
File Written to text.001.txt

Orignal Summary:  Good taste

Text
  Word Ids:    [113, 626, 1687, 68, 35, 1644, 1519, 1739, 0, 800, 2162, 1519, 1739, 1375, 24, 471]
  Input Words: like slight pineapple flavor better plain coconut water good well chilled coconut water supposed healthy drink

Summary
  Word Ids:       [15, 54]
File Written to text.A.001.txt
  Response Words:  nice taste
Time:  22.70325779914856
```

Fig A3.6 Response for input text

Fig A3.7 Precision, Recall, Rouge value

REFERENCES

[1] L.D.S. Cabral, R.D. Lins and R.F. Mello (2014) 'A platform for language independent summarization', In Proceedings of the 2014 ACM Symposium on Document Engineering, pp.203-206.

[2] M. Gambhir and V. Gupta (2016) 'Recent automatic text summarization techniques: A survey', Artif. Intell. Rev., pp.1-66.

[3] D. Ciresan, U. Meier and J. Schimdhuber (2012) 'Multi-column deep neural networks for image classification', IEEE conference on Computer Vision and Pattern Recognition, pp.3642-3649.

[4] Goldberg and Yoav 'A Primer on Neural Network Models for Natural Language Processing', Journal of Artificial Intelligence Research, pp. 45–420.

[5] Mani 'Automatic Summarization', John Benjamins Publishing, Amsterdam, ISBN-10:90272495865, pp.285.

[6] S. Polsley, P. Jhunjhunwala and R. Huang (2016) 'Casesummarizer: A system for automated summarization of legal texts', In Proceedings of COLING 2016, 26t International Conference on Computational Linguistics-System Demonstrations, pp.258-262.

[7] Y.C. Chen and M. Bansal 'Fast abstractive summarization with reinforce-selected sentence rewriting', arXiv preprint arXiv:1805.11080

[8] Abeer Alzuhair and Mohammed Al-Dhelaan (2019) 'An approach for combing multiple weighting schemes and ranking methods in Graph-based Multi-Document Summarization', Vol.7.

[9] Begum Mutlu, Ebru A.Sezer and M.Ali Akcayol (2019) 'Multi document extractive text summarization: A comparative assessment on features', pp.1-13.

[10] Rupal Bhargava, Yashvardhan Sharma and Gargi Sharma (2016) ‘ATSSI: Abstractive Text Summarization using Sentiment Infusion’, Twelfth International Multi-conference on Information Processing, India.

[11] G Padma Priya and K Duraisamy (2014) ‘An Approach for Text Summarization using Deep learning Algorithm’, Jounal of Computer Science, pp.1-9.

[12] Khosrow Kaikhah ‘Text summarization using neural networks’, Department of computer science, Texas State University.

[13] P Genest and G Lapalme (2012) ‘Fully Abstractive Approach to Guided Summarization’, Proceedings of the 50th annual meeting of the association for Computational Linguistics, Canada, pp. 354-358.

[14] Barzilay R and K R McKeown (2005) ‘Sentence Fusion for Multi Document News Summarization’, Computational Linguistics, pp.297-327.

[15] G. Dineshnath and S. Saraswathi (2018) ‘Comprehensive survey for abstractive text summarization’, International journal of Innovations and Advancement in Computer science, pp.215-219.

[16] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Ben (2014) ‘Neural machine translation by jointly learning to align and translate’, CoRR, abs/1409.0473.

[17] Minh-Thang Luong, Hieu Pham and Christopher D. Manning (2015) ‘Effective approaches to attention based neural machine translation’, In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp.1412–1421.

[18] Manjula Subramaniam and Prof. Vipul Dalal (2015) ‘Test Model for Rich Sematic Graph Representation for Hindi Text using Abstractive Method’, Vol.2, No.2, e-ISSN: 2395-0056, p-ISSN: 2395 0072.

[19] Jagadish S Kallimani, K G Srinivasa and B Eswara Reddy (2014) ‘A Comprehensive Analysis of Guided Abstractive Summarization’, International Journal of Computer Science Issues, Vol.11, No.1, e-ISSN: 1694-0784, p-ISSN: 1694-0814.

[20] Baralis, E Cagliero, L Jabeena and S Fiori, Shah S (2013) ‘Multi Document Summarization based on the Yago Ontology’, Expert Systems with Applications, pp.6976-6984.

[21] L Fathy, D Fadl and M Aref (2012) ‘Rich Semantic Representation based approach for Text Generation’, The Eight International Conference on Informatics and Systems (INFOS), Egypt, pp. 19-27.

[22] E.Lloret and M palomar (2011) ‘Analysing the use of Word Graphs for Abstractive Text Summarization’, Proceedings of the 1st International Conference on Advances in Information Mining and Management, Barcelona, Spain, pp.61-66.

[23] I F Moawad and M Aref (2012) ‘Semantic Graph Reduction Approach for Abstractive Text Summarizaton’, IEEE International Conference on Natural, Egypt, pp.132-138.

LIST OF PUBLICATIONS

- Dr.J.Anitha, M.Raahavi, M.Rehapriadarsini, S.S.Sudarshana (2020) ‘Abstractive Text Summarization’,Journal of Xidian University, Vol.14, Issue 6, pp. 854-857