Extractive Text Summarization with Pre-training of deep Bidirectional Encoder Representations from Transformers and Clustering

A Thesis

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Robiul Hasan Nowshad	**160104061**
Samia Zahan	**160104057**
Muna Saha	**160104060**
Tasmia-Tuj-Juha	**160104095**

Supervised by

Mr. Tanveer Ahmed Belal

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

November 2020

CANDIDATES’ DECLARATION

We, hereby, declare that the Thesispresented in this report is the outcome of the investigation performed by us under the supervision of Mr. Tanveer Ahmed Belal , Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE400: Project and Thesis I and CSE450: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this Thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Robiul Hasan Nowshad
160104061

Samia Zahan
160104057

Muna Saha
160104060

Tasmia-Tuj-Juha
160104095

CERTIFICATION

This Thesis titled, **“Extractive Text Summarization with Pre-training of deep Bidirectional Encoder Representations from Transformers and Clustering”**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in November 2020.

Group Members:

Robiul Hasan Nowshad	**160104061**
Samia Zahan	**160104057**
Muna Saha	**160104060**
Tasmia-Tuj-Juha	**160104095**

Mr. Tanveer Ahmed Belal
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Professor Dr. Kazi A. Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

First and foremost, we are grateful to Almighty Allah for blessing us with the good health and well-being we required to work on this thesis. Next, we would like to express our utmost gratitude to our respected supervisor Mr. Tanveer Ahmed Belal for his valuable guidance throughout the planning and development of our undergraduate thesis. His supervision and motivation has helped us all the time and he always encourage us to gather knowledge and try different things. We are truly grateful for having him as our supervisor and mentor for our undergraduate thesis.

We also want to take this opportunity to thank Dr. Kazi A. Kalpoma, head of the department and all our respected faculty members of the department of CSE, AUST for their help in offering us some useful resources and precious advice. Without them we could not complete this research study. Last but not least, we want to thank the people who made us come this far, our parents, for their constant support, blessings and unconditional love which helped us to progress. We would like to pay our warmest tribute to our friends for being there for us and helped us regarding different problems and mental support.

Dhaka
November 2020

Robiul Hasan Nowshad

Samia Zahan

Muna Saha

Tasmia-Tuj-Juha

ABSTRACT

The quantity of information on the internet is massively being increased and getting gigantic day by day. Our life is too limited and synchronizing with this limited time is very important. To cope up with the gigantic amount of data within the shortest possible time is the greatest challenge now. With such a big amount of data circulating in the digital space, there is a vast necessity to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages. Text summarizing is a technique of creating a short, accurate, and fluent summary of a longer text document. Automatic text summarizing is a thick-headed topic of Natural Language Processing and Machine Learning. There exist two approaches of automatic text summarizing, namely: - I) Extractive text summarization and II) Abstractive text summarization. We have focused on Extractive text summarization in our research work. For data representation, weighting and grouping we have used a conceptually simple and empirically powerful model named BERT (Bidirectional Encoder Representations from Transformers) for text embedding and K-Means clustering algorithm to identify sentences closest to the centroid for summary selection. For finding true value of k we've emphasized on Elbow method. The evaluation is done by using ROUGE .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Textual information is instantaneously growing and spreading speedily over the internet. As it is accessible by any ordinary user from anywhere of the world who can generate, infuse, or acquire information mostly in textual form. So the growth is quite natural and simultaneously a new challenge arises which is acquiring the specific desired knowledge among the huge amount of information that a user is searching for. It would be much easier for the users if they are provided with a shorter version of the information which will reflect a vast knowledge in a summarized form. As one of the methods to tackle this challenge, text summarization process diminishes the redundant information and retrieves the useful and relevant information from a text document to form a compressed and shorter version which is easy to understand and time saving while reflecting the main idea of the discussed topic within the document. It is also a fact that manually or man-made summarization has the possibility of being biased and the occupied time for a human made summarization is also a crucial fact. Text Summarization is one of those applications of Natural Language Processing (NLP) which is bound to have a huge impact on our lives. With growing digital media and ever growing publishing people do not have the time to go through entire articles / documents / books to decide whether they are useful or not. Thankfully, the technology of automatic text summarization is here. So, instead of studying every related document to decide if the document is beneficial or not to read can be decided through text summarization, just by reading the summary generated from the document while wasting less time. Except for literature and culture fields, every field demands the necessity of text data summarization. Though lots of institutions, organizations and also individual intelligence are already working on the field but there is still a large opportunity to work on this field to achieve much better accuracy which is the reason why we are keen to work on this topic, automatic text summarization.

1.2 Automatic Text Summarization

Summarization is the task of concentrating a piece of text to a brief version, reducing the size of the initial text while preserving the mainstay as well as the meaning of content at the same time. The main intention of a text summarization is to create a coherent and fluent summary having only the main points outlined in the document. Automatic Text Summarization is one of the most challenging and interesting problems in the field of Machine Learning (ML) and Natural Language Processing (NLP).The current developments in Automatic text Summarization are owed to research into this field since the 1950s when Hans Peter Luhn's paper titled "The automatic creation of literature abstracts" was published. Automatic text summarization is a process of generating a concise and meaningful summary of text from multiple text resources such as books, news articles, blog posts, research papers, emails, and tweets with the help of an automated system. The demand for automatic text summarization systems is spiking these days because of the availability of large amounts of textual data. There are important applications for text summarization in various NLP related tasks such as text classification, question answering, legal texts summarization, news summarization, and headline generation. Moreover, the generation of summaries can be integrated into these systems as an intermediate stage which helps to reduce the length of the document.

1.3 Motivation

Propelled by the modern technological innovations, data is to this century what oil was to the previous one. With such a big amount of data circulating in the digital space, there is a need to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages. Furthermore, applying text summarization reduces reading time, accelerates the process of researching for information, and increases the amount of information that can fit in an area. Since manual text summarization is a time expensive and generally laborious task, the automatization of the task is gaining increasing popularity and therefore constitutes a strong motivation for academic research. Today, our world is parachuted by the gathering and dissemination of huge amounts of data. In fact, the International Data Corporation (IDC) projects that the total amount of digital data circulating annually around the world would sprout from 4.4 zettabytes in 2013 to hit 180 zettabytes in 2025. That's a lot of data! In the big data era, there has been an explosion in the amount of text data from a variety of sources. This volume of text is an inestimable source of information and knowledge which needs to be effectively summarized to be useful. This increasing availability of documents has demanded exhaustive research in the NLP area for automatic text summarization. Automatic text summarization is the task of producing a concise and fluent summary without any human help while preserving the meaning of the original text document. It is very challenging, because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task. Various models based on machine learning have been proposed for this task. Most of these approaches model this problem as a classification problem which outputs whether to include a sentence in the summary or not. Other approaches have used topic information, Latent Semantic Analysis (LSA), Sequence to Sequence models, Reinforcement Learning and Adversarial processes.

1.4 Objective

The main objective is to produce an efficient summary with an automated system without changing the concept of the document. Here we have focused experimenting for extractive based text summarization using BERT and unsupervised learning which is K-Means Clustering. A raw dataset as the system input will not be recognized by a machine. The pre-processing step will tokenize the incoming paragraph text into clean sentences and pass the tokenized sentences to the BERT model for inference to output embedding. The embedding will be clustered with the K-Means algorithm. By selecting the embedded sentences that were closest to the centroid as the candidate summary sentences, the summary will be provided, finally. BERT is a multi-layer bidirectional Transformer encoder designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. BERT has been pre-trained on two tasks: (i) Masked Language Model (MLM) and (ii) Next Sentence Prediction. In MLM instead of predicting every next token, a percentage of input tokens is masked at random and only those masked tokens are predicted. Next sentence prediction task is a binary classification task in which, given a pair of sentences, it is predicted if the second sentence is the actual next sentence of the first sentence. After embedding using BERT for clustering, an efficient value of K will be obtained by Elbow method.

So, the overview of our summarization service engine is it holds a pipeline which tokenized the incoming paragraph text into clean sentences, passed the tokenized sentences to the BERT model for inference to output embedding, and then clustered the embedding with K-Means, selecting the embedded sentences that were closest to the centroid as the candidate summary sentences. The objective of our experiment was optimization of topic coverage and optimization of readability. We wanted to explore ways to generate a summary which should cover the whole topic of the documents. The generated summary should have the most relevant information of the source documents. The generated summary will also be balanced and won't have lack of cohesion which will be satisfactorily readable.

Chapter 2

Approaches for Text Summarization

A summary can have variants in its formation. It may point to some important parts of the original document which is an indicative way or it may cover all the information of text that is relevant to the input document which is an informative way. Summary can be generated automatically by following certain methods and the procedures are expected to provide some advantages like: (i) The Summary size should be controllable (ii) Contents of the summary are to be deterministic (iii) The sentence (text element) in the summary and sentence position in the original input document should be inked up and the link is allowed to establish earlier.

2.1 Text Summarization Approaches

Classification of automatic text summarization can be dependent on many facts. An automatic summarization can entail a combined human and software effort. It can also be classified on the basis of Single vs. Multi-document and indicative vs. informative. Document length and type of the input text also heavily influences the sort of summarization approach. But there are primarily two main approaches to automatic summarization of text in NLP which are:

1. Abstraction-based Summarization
2. Extraction-based Summarization

2.1.1 Abstraction-based Summarization

Abstraction-based summarization method uses advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text.

The abstraction technique entails paraphrasing and shortening parts of the source document. When abstraction is applied for text summarization in deep learning problems, it can overcome the grammar inconsistencies of the extractive method. The abstractive text summarization algorithms create new phrases and sentences that relay the most useful information from the original text, just like humans do. Abstractive summarization methods aim at producing summary by interpreting the text using advanced natural language techniques in order to generate a new shorter text where parts of which may not appear as part of the original document, that conveys the most critical information from the original text, requiring rephrasing sentences and incorporating information from full text to generate summaries such as a human-written abstract usually does. In fact, an acceptable abstractive summary covers core information in the input and is linguistically fluent. Thus, they are not restricted to simply selecting and rearranging passages from the original text. In case of abstractive text summarization as it more closely reproduces human summarization in that it uses a vocabulary beyond the specified text, abstracts key points, and is generally smaller in size (Genest Lapalme, 2011). However, the text summarization algorithms required to do abstraction are more difficult to develop; that's why the use of extraction is still popular. In general, building abstract summaries is a challenging task, which is relatively harder than data-driven approaches such as sentence extraction and involves complex language modelling. While this approach is highly desirable and has been the subject of many research papers but since it imitates how humans summarize a material, it is difficult to produce automatically, either requiring several GPUs to train over many days for deep learning or complex algorithms and rules with limited generalized for traditional NLP.

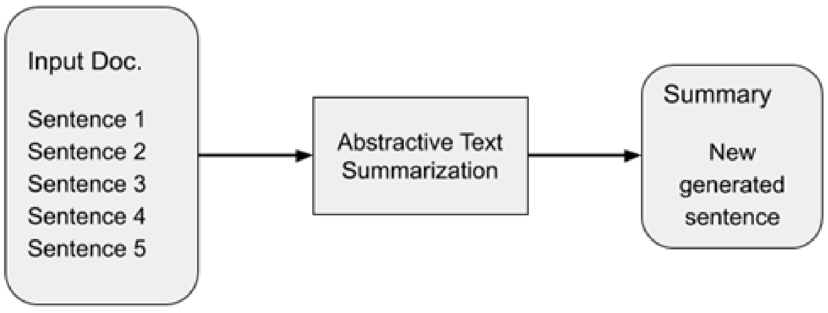

Figure 2.1: Abstractive Summarization

2.1.2 Extraction-based Summarization

Extraction-based summarization method relies on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method. Extractive summarization generally utilizes the raw structures, sentences, or phrases of the text and outputs a summarization, influencing only the content from the source material. Extractive summarization picks up sentences directly from the document based on a scoring function to form a coherent summary. This method works by identifying important sections of the text cropping out and stitching together portions of the content to produce a condensed version. Basically, the extractive text summarization technique involves pulling key phrases from the source document and combining them to make a summary. The extraction is made according to the defined metric without making any changes to the texts. As the extractive type refers to the process in which we focus on selecting meaningful information from or paragraph within the primary document and link them in shorter form and While for the abstractive type, it is required to understand the elemental image discussed in the document and it precisely explains that new concept using a fair and clear natural language. Here, we focused on extraction-based summarization. This summarization technique largely applies for domain specific documents like Clinical Trial Descriptions, Criminal History for police work and so on. There are many approaches for extractive text summarization like, unsupervised learning, semi-supervised learning, supervised learning, neural networks, using fuzzy logic, deep learning etc.

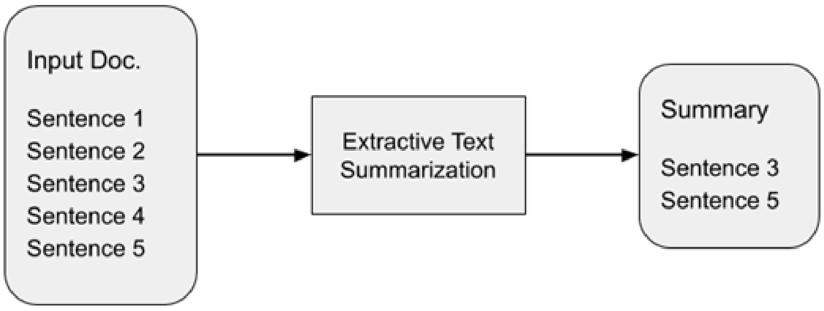

Figure 2.2: Extractive Summarization

2.2 Research on Automatic Text Summarization

In the past, extractive summarizers have been mostly based on scoring sentences in the source document. The most common and recent text summarization techniques use either statistical approaches, or linguistic techniques. The high frequency words, standard keyword, Cue Method, Title Method, and Location Method are used for weighting the sentences. Most of the current automated text summarization systems use extraction methods to produce a summary. [13]

Christian Guldena et al. [1] generated a corpus consisting of the detailed description and brief summary of each trial they worked on the entire public clinicaltral.gov study record database. After cleaning the raw data, to identify semantically valid detail summary pairs, overlapping semantics were measured by Basic text similarity metrics, which used Cosine distance and the weighting was done by term frequency inverse document frequency (TF-IDF) weighing method. They used a python sumy library to generate and also considered the performance of TexRank , LSA, Luhn, SumBasic and KLSum algorithms. They achieved 74sentence-level reduction. The ROUGE scores: ROUGE-1(0.38), ROUGE-2(0.17), and ROUGE-L (0.33).

Rahim Khan et al. [2] also used the TF-IDF score weight after the basic preprocessing phase. But the summarization was done by choosing clusters having higher priority. While clustering (organizing text documents in different groups according to their semantic similarity) they focused on the proper number of clusters that is the true K value. K value was measured by the Elbow/Silhouette method. For different K values the experiment shows different levels of accuracy. The BLEU scores: BLEU-1(0.69345), BLEU-2(0.458992).

Sean MacAvaney et al. [3] authors used standard neural approaches for abstractive summarization following the sequence to sequence framework. Where an encoder network reads the input and a separate network learns to generate the summary. They used Bidirectional LSTMS (BiLSTMs) as encoder - decoder and PG (pointer generator) model which incorporates a copy mechanism to directly copy text from input where appropriate. They also compared their result with the well established extractive methods and state-of-the-art abstractive summarization, achieved a great accuracy. The ROUGE scores: ROUGE-1(37.64(36.33

Feng Li et al. [4] build a summarization system using Extraction-based method. They use single document summarization and keyword based approach. They extract keywords from input documents then extract similar documents from the past which they created as external corpus of past news articles. From those articles they discover new keywords and expand them to create summary. But it's not efficient enough for total topic articles. The ROUGE scores: ROUGE-1(0.772), ROUGE-2(0.620), and ROUGE-L (0.580).

Changjian Fang et al. [5] authors created a novel word-sentence co-ranking model named CoRank for automatic extractive text summarization. CoRank combines the word sentence relationship with the graph- based ranking model. The ROUGE scores: ROUGE-1(0 .697), ROUGE-2(0.606), ROUGE-L (0.661).

Yulia Ledeneva et al. [6] used TF-IDF extraction based summarization method .After doing the basic pre-processing the TF-IDF score is weighted by term selection, term weighting, sentence clustering and sentences selection respectively. Sentences are selected using an unsupervised algorithm, particularly K-means algorithm, for discovering the groups of sentences with similar meaning. For measuring the similarity between two sentences the Euclidean distance is used here. The ROUGE scores: ROUGE-1 (0.44), ROUGE-2 (0.43), ROUGE-3 (0.44).

Mohammad Reza Keyvanpour et al [7], used extraction based summarization on text has been done using neural networks. A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks consist of input layer, hidden layer, output layer. Artificial Neural Network (ANN), Restricted Boltzmann machines (RBMs) are studied here which are referred to as shallow models. Recurrent Neural Network (RNN) is applied in which nodes are connected to each other in form of a directed graph along a sequence. Convolutional Neural Network (CNN) also is known as feed-forward artificial neural networks consisting of several layers including input layer, hidden layers which are a combination of convolution, pooling and fully connected which is the output layer. They ROUGE scores in M for medium and H for high.

Yulia Ledeneva et al. [8] built a summarization system using Extraction based summarization method using Frequent Sequences (FSs). They analyse several options for simple language-independent statistical term selection and corresponding term weighting, based on units larger than one word and also show that maximal frequent sequences (MFSs), as well as single words that are part of bigrams repeated more than once in the text, are good terms to describe documents. They followed 4 steps which are (a) Term selection (b) Term weighting (c) Sentence weighting (d) Sentence selection. They used a ROUGE evaluation toolkit which uses n-gram statistics. Basically they used an evaluation method to compare the usefulness and truthfulness of the summary. They achieved ROUGE scores 0.44, 0.45 and 0.47 on different terms.

J.N.Madhuri et al. [9] worked on the extractive summarization, the summarizer takes input as text file and tokenization of an input text is done to find the terms of the text. Then stop words are removed to filter the text. Finally, a part-of speech tag is added to each token. Weights (Wt) are assigned to individual tokens. Maximum weight of the token is considered through the Weight Term Frequency (WTF). Their highest ROUGE scores are ROUGE 1 - 0.77, ROUGE 2 - 0.78.

Xiangke Mao et al. [10] presented three methods (i) Linear combination method (ii) LexRank (iii) Biased-LexRank, combining supervised learning with unsupervised learning to generate a single document summary. The first method combines the scores of two methods linearly. The second one regards the scores of unsupervised methods as a feature of supervised learning. The third one regards the scores of supervised learning as a priori value of nodes in the graph model. When using these three methods to score sentences, the statistical characteristics of sentences and the relationship between sentences are integrated, which can evaluate the importance of sentences in documents more comprehensively, thus improving the accuracy of sentence scoring.

Derek Miller [23] reports on the project called “lecture summarization service” which is a python-based RESTful service that utilizes the BERT model for text embeddings and K-

Means clustering to identify sentences closest to the centroid for summary selection. The purpose of the service was to provide students a utility that could summarize lecture content, based on their desired number of sentences. The results of utilizing BERT for extractive text summarization were promising excluding some areas where the model struggled.

Chapter 3

Methodology

3.1 Problem State

For extractive text summarization, here our main goal is to select the most important sentences (i.e. sentences that portray the main context) from a given document. To achieve this goal, we construct a strategy with the help of deep learning and machine learning and some frameworks like NLTK, Pytorch, Tensorflow.

We divide our problem into 3 sub-problems.

1. Perceiving quality sentence embedding representation
2. Finding similar sentences as cluster
3. Extract summary from clusters

Our approach to solving these sub-problems one by one is given below.

3.2 Methodology

To illustrate the comprehensive workflow of the algorithm that we've implemented for the purpose of our main objectives, a diagram is provided below fig 3.2.1 which shows the whole process from the start to the end. Here we divided the whole process into some small steps which will be explained in the next chapters. Based on the input document Text summarization is basically of two types. It may work with single document or multi document. Our system can deal with both. The first step of our methodology is to take a single or multi document which we want to summarize as an input. But human language is not understandable from machines. It needs conversion and processing. So, after reading

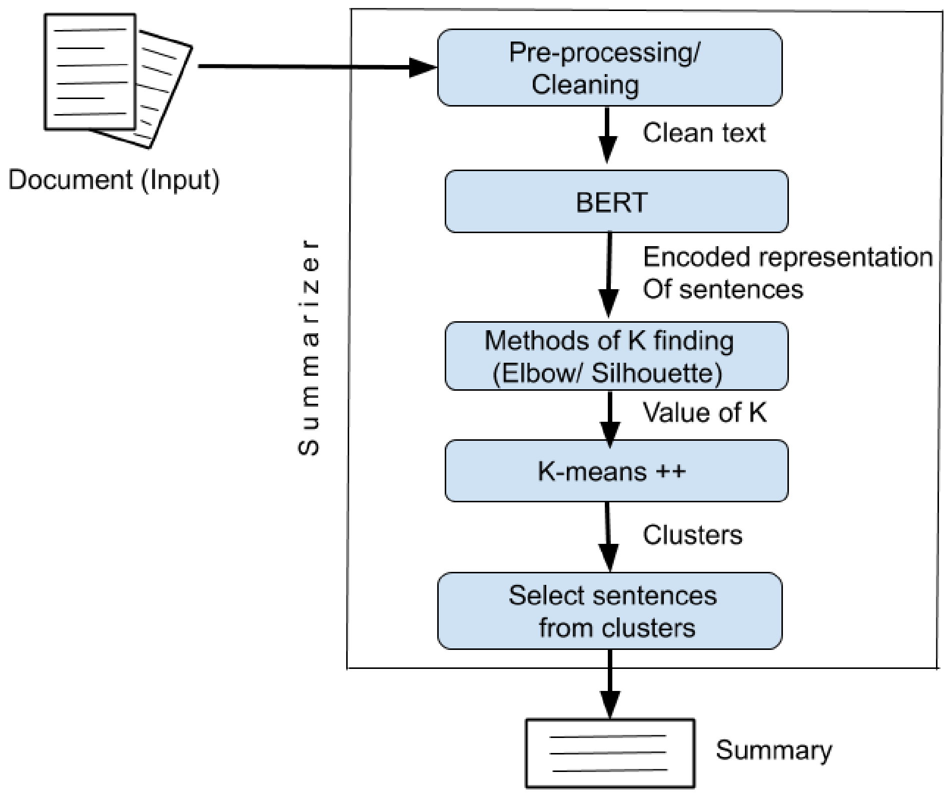
Document (Input)
Summarizer
Pre-processing/ Cleaning
Clean text
BERT
Encoded representation Of sentences
Methods of K finding (Elbow/ Silhouette)
Value of K
K-means ++
Clusters
Select sentences from clusters
Summary

Figure 3.1: Methodology

the document, there is a function in our system called pre-processing which helps to clean up the document from unwanted words and characters. Over all it filters all the information within the input document which carries negligible impact on the main context. This is done with the help of regular expression, stemming, lemmatization. And it is the initial process for every NLP (Natural Language Processing). Second step is converting the text data into numerical format so that the machine can understand and interact with the data. This is the featuring step. The featuring of the document can be done by several processes stated earlier. Our system is using Bidirectional Encoder Representations from Transformers (BERT) models for this purpose. The BERT converts the data into vectors paying much more attention to the semantic meaning of the word. It works comparatively far better then the other contemporary methods like: TF-IDF, Bag of Words, Word2Vec, GloVe etc. After finishing the step we have numeric representation for our document which we can feed any machine learning algorithm. The next task is to group the similar sentences; that's the sentences which carry approximately similar meaning for the document. There are a lot of approaches for this clustering purpose. Here we've used unsupervised learning K-Means Clustering for the task. But before feeding the data to the K-Means Clustering algorithm, some prior values like the value of k (number of clusters), initial seed or centroids for that k number of clusters, learning rate etc need to be specified. In basic K-means the initial centroids are selected randomly. Though the random selection may drive to the local optima sometime. So tuning the initial seed value can give better results. This issue will be discussed later in the K-means section. Apart from that our system needs to find appropriate value for K also. To find efficient value for K, we've used 'Elbow Method' also 'Silhouette Method' is used for this. In step three we feed the data to the Elbow Method and The Elbow Method shows us the elbow graph from which we choose the elbow point, which is the value of k. In step four, we pass the value for K which we get from step three and pass the data to K-means Clustering. It clusters the sentences in k number of clusters. In step five, we select sentences from each cluster nearest to the centroids that is the most relevant sentence that carries the context of that cluster. To ensure the rational presence of information under each sub-context/ cluster we will choose more number of sentences from the cluster having higher frequency and less number of sentences from the clusters having lower frequency. We can determine the number of sentences to be picked up from each cluster maintaining the ratio in several ways. In our paper we just take the ceiling value of the ratio of size of each cluster and the size of the smallest cluster.

Chapter 4

Data Acquisition and Preprocessing

For our research, our first step was collecting a dataset. To summarize the document and evaluation two things are needed. First thing is the documents which we have to summarize by our automated summarizing system and the second thing is reference summaries of those documents (i.e. human made summaries) for the evaluation to find out how good our algorithm is. After the data acquisition phase the preceding step is the pre-processing step, which is needed to refine the data and prepare it worthy for the summarization system to work on it.

4.1 Data Acquisition

For our algorithm we've collected our dataset from [14]. We selected BBC news articles dataset which consists of 2225 documents from the BBC news website corresponding to stories in five topical areas like- politics, sports, technology, business, entertainment from 2004-2005 in the News Articles folder. For each article, five summaries are provided in the Summaries folder. The first clause of the text of articles is the respective title. Evaluating the algorithm we needed reference summaries, for that we use the summaries from the Summaries folder. That's all to fulfil our requirement.

4.2 Data Pre-processing and Cleaning

Pre-processing is the primary step required for all Natural Language Processing. It is the process of preparing the data in a readable format. It cleans the data and reduces the noise. We applied basic pre-processing like tokenization, filtering, stemming or lemmatization and stop-word removal.

1. **Tokenization:** Tokenization is a very common task in NLP, it is basically a task of chopping a character into pieces, called a token. On our dataset we applied tokenization using python NLTK library. From NLTK we use a tokenization function. One sentence tokenizer which takes documents as input and return list of sentences. Two word tokenizer which takes a sentence as input and return list of words.

2. **Filtering:** In documents, there are some words, characters which have very low value, sometimes no value for text summarizing, which should be removed from the text document. Here we used a python regular expression library known as regex which helped us to remove non-word characters, extra-spaces and other unnecessary things which we don't need for summarizing.

3. **Stemming:** Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP). Basically, the vital goal of the stemming process is to change the words to its root words. But there is a drawback of the stemming process, like it sometimes changes the meaning and sometimes creates words for finding the root word which has no meaning. So to find root words we didn't use it in our algorithms.

4. **Lemmatization:** Lemmatization (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form. In computational linguistics, lemmatization is the algorithmic process of determining the lemma of a word based on its intended meaning. Unlike stemming, lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighbouring sentences or even an entire document. In our algorithm, for pre-processing we used NLTK lemmatization for finding the root word.

5. **Stop-Word Removal:** The words like preposition, articles, conjunction such as is, the, an, a, when, and, or, but, or the non-informative words and certain high frequency words are the "stop-words". The stop-word removal technique is the process of expelling them from the document. Because they do not give any meaning and are less significant.

Chapter 5

Data Representation

5.1 Introduction

Data representation is the process of transforming the data into machine understandable as in form of vectors or numbers. There are a lot of process for transforming the text data into machine understandable form, Such as Bag of words model, TF-IDF model, 1-Hot Encoding Model, N-gram Model, (these models represent the text data in matrix form) while the Word to Vector model, Vector Semantics represent the data in vector form. But for sentence representation they are not good enough. Like TF-IDF gives us a statistical representation of data. It's not a good enough representation of text. Same way Word2vec, GloVe just encodes the word, doesn't keep the context. On the other hand, the recent BERT model encodes documents giving attention to the context.

5.2 Moving Towards Deep Learning

Until recently (2017), the recurrent neural network (RNN) was the default approach for many natural language processing applications, requiring massive amounts of data, expensive compute resources, and several hours of training to achieve acceptable results, while suffering from poor performance with very long sequences and was prone to over fit. RNN deals with sequence data (input has some defined ordering). While back propagating it produces gradients that vanish or explode if the sequence is too long. So to solve the issue LSTM (long short term memory) is used. An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells. These operations are used to allow the LSTM to keep or forget information. LSTMs solves the vanishing gradient problem using a unique additive gradient structure that includes direct access to the forget gate's activations, enabling the network to encourage desired behavior from the error gradient using frequent gates update on every time step of the learning process. But LSTMs only solve gradient vanishing problems so they still suffer from gradient explosion problems.

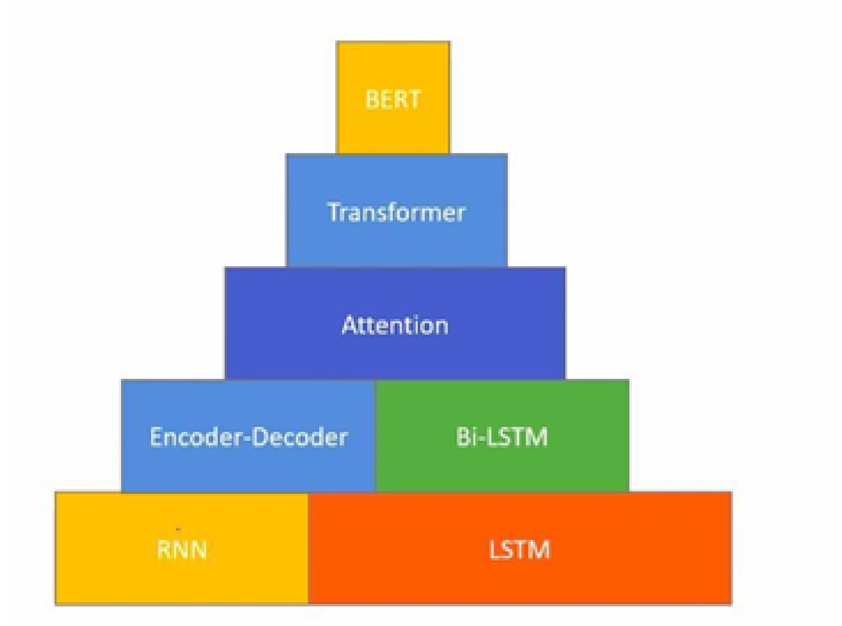

Figure 5.1: Seq2Seq Pyramid

Another problem with both RNNs and LSTMs is, they can't do the task in parallel. That is, in the case of a language translation model, each word is sent to the network sequentially. The time it takes to calculate something would really depend on the overall length of the text since each word is given as the input to the network. To solve these problems (explosion of gradient, parallelize sequential data) Transformer neural network architecture is used which has been introduced by (Vaswani, et al., 2017).

5.3 Transformer

Transformer is a model that uses attention to boost the speed. More specifically, it uses self-attention. The Transformer consists of a set of encoders and decoders. The encoders are very similar to each other. All encoders have the same architecture. Decoders also share the same property, i.e. they are very similar to each other. Each encoder consists of two layers: Self-attention and a feed Forward Neural Network. The encoder's inputs first flow through a self-attention layer. It helps the encoder look at other words in the input sentence as it encodes a specific word. The decoder has both those layers (self attention, feed forward neural network), but between them there is an encoder-decoder attention layer that helps the decoder focus on relevant parts of the input sentence. The Encoders and Decoders are stacked on top of each other multiple times, which is described by Nx in the figure. The

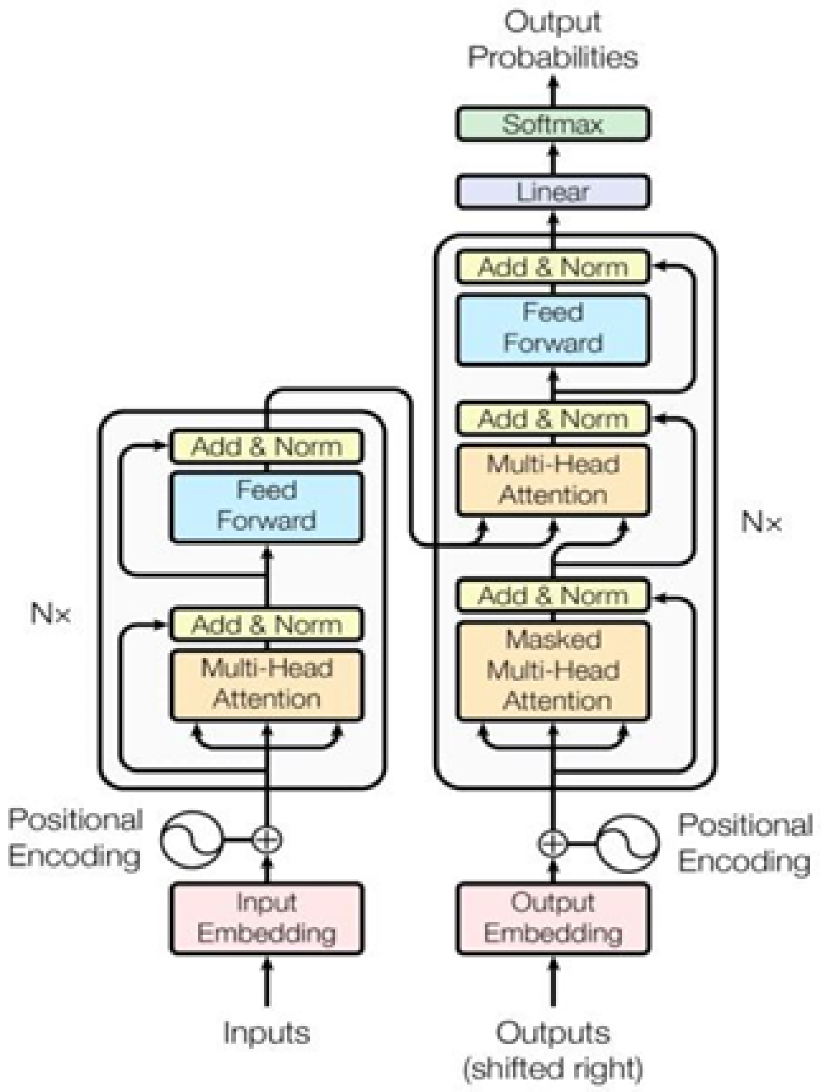

Figure 5.2: Transformer Model

inputs and outputs are first embedded into an unidimensional space since strings cannot be used directly. One important part of the model is the positional encoding of the different words. Since there are no recurrent networks that can remember how sequences are fed into a model, so every word/part in a given sequence should have a relative position as a sequence depends on the order of its elements. These positions are added to the embedded representation (n-dimensional vector) of each word. Let's have a closer look at these Multi-Head Attention bricks in the model:

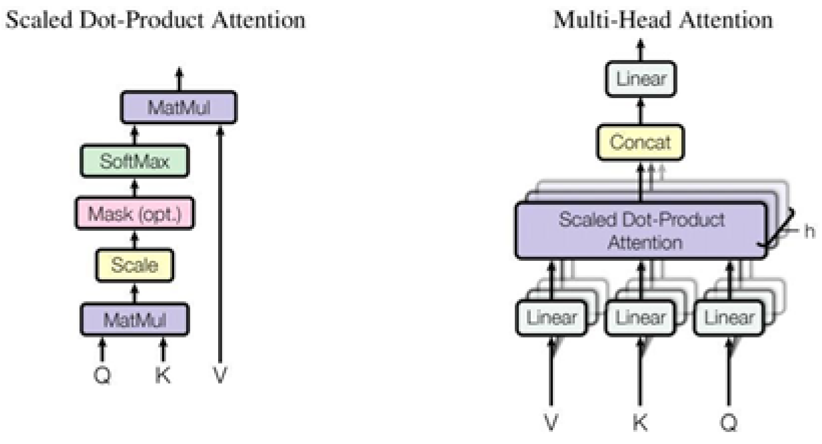

Figure 5.3: Dot Product and Multi-Head Attention

The left description of the attention-mechanism can be described by the following equation:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Q is a matrix that contains the query (vector representation of one word in the sequence), K are all the keys (vector representations of all the words in the sequence) and V are the values, which are again the vector representations of all the words in the sequence. For the encoder and decoder, multi-head attention modules, V consists of the same word sequence as Q. However, for the attention module that is taking into account the encoder and the decoder sequences, V is different from the sequence represented by Q.

To simplify this a little bit, we could say that the values in V are multiplied and summed with some attention-weights a, where our weights are defined by:

$$a = softmax(\frac{QK^T}{\sqrt{d_k}})$$

This means that the weights a are defined by how each word of the sequence (represented by Q) is influenced by all the other words in the sequence (represented by K). Additionally, the SoftMax function is applied to the weights to have a distribution between 0 and 1. Those weights are then applied to all the words in the sequence that are introduced in V (same vectors than Q for encoder and decoder but different for the module that has encoder and decoder inputs).

The right hand picture describes how this attention-mechanism can be parallelized into multiple mechanisms that can be used side by side. The attention mechanism is repeated multiple times with linear projections of Q, K and V. This allows the system to learn from different representations of Q, K and V, which is beneficial to the model. These linear representations are done by multiplying Q, K and V by weight matrices W that are learned during the training.

Those matrices Q, K and V are different for each position of the attention modules in the structure depending on whether they are in the encoder, decoder or in-between encoder and decoder. The reason is that we want to attend to either the whole encoder input sequence or a part of the decoder input sequence. The multi-head attention module that connects the encoder and decoder will make sure that the encoder input-sequence is taken into account together with the decoder input-sequence up to a given position. After the multi-attention heads in both the encoder and decoder, we have a pointwise feed-forward layer. This little feed-forward network has identical parameters for each position, which can be described as a separate, identical linear transformation of each element from the given sequence.

5.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language [24]. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectional trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

5.4.1 BERT Working Procedure

ERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in [24] by Google.

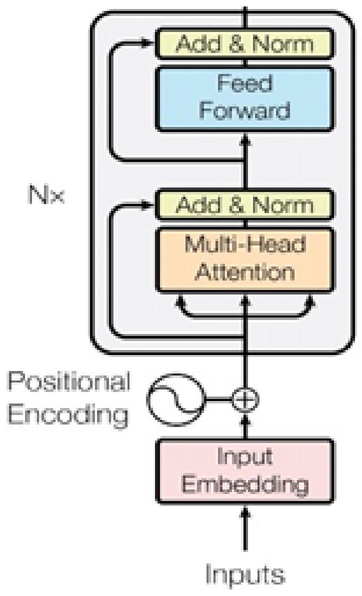

Figure 5.4: Encoder from Transformer

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The figure above is a high-level description of the Transformer encoder. The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

When training language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence (e.g. "The child came home from —–"), a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies: Masked Language Model (MLM) and Next Sentence Prediction.

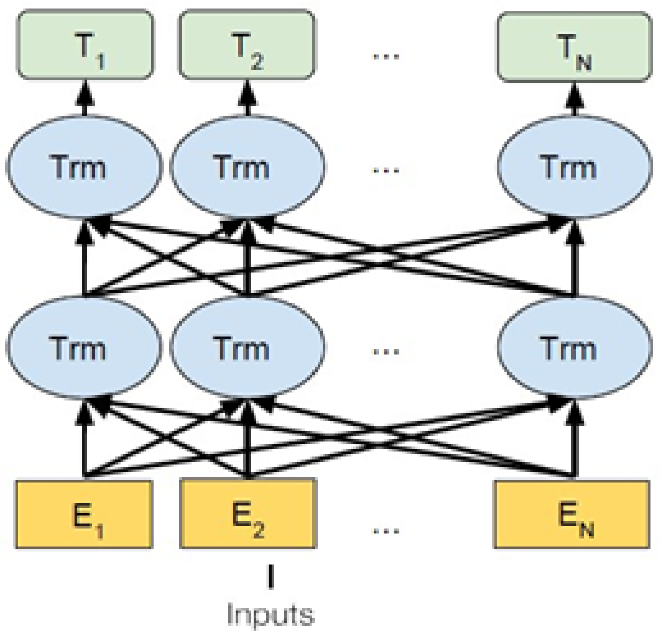

Figure 5.5: Bidirectional Computation

5.4.2 Masked Language Model (MLM)

Before feeding word sequences into BERT, 15replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires: 1) Adding a classification layer on top of the encoder output. 2) Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension. 3) Calculating the probability of each word in the vocabulary with softmax.

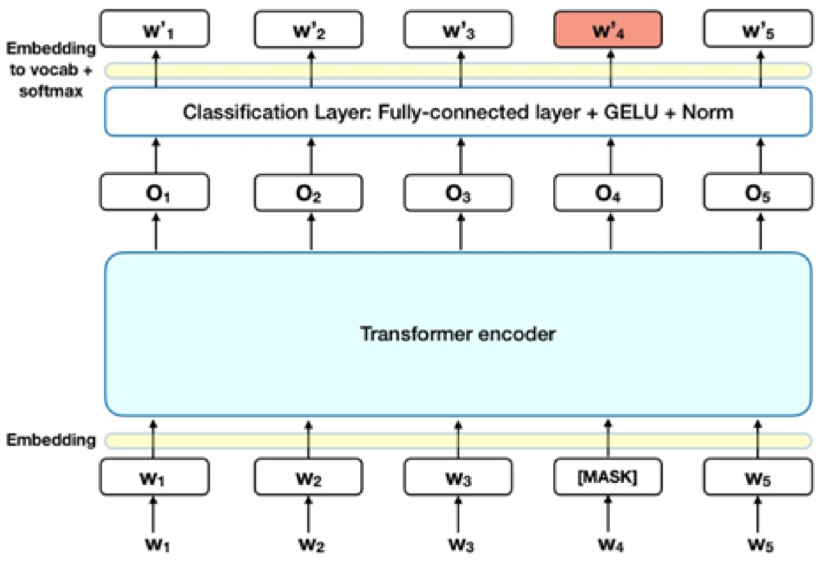
Embedding to vocab + softmax
w'1
w'2
w'3
w'4
w'5
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 5.6: BERT embedding layer

The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words. As a consequence, the model converges slower than directional models, a characteristic which is offset by its increased context awareness. BERT's bidirectional approach (MLM) converges slower than left-to right approaches (because only 15but bidirectional training still outperforms left-to-right training after a small number of pre-training steps. In practice, the BERT implementation is slightly more elaborate and doesn't replace all of the 15

5.4.3 Next Sentence Prediction (NSP)

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50the subsequent sentence in the original document, while in the other 50sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model: 1) A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence. 2) A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embedding with a vocabulary of 2. 3) A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

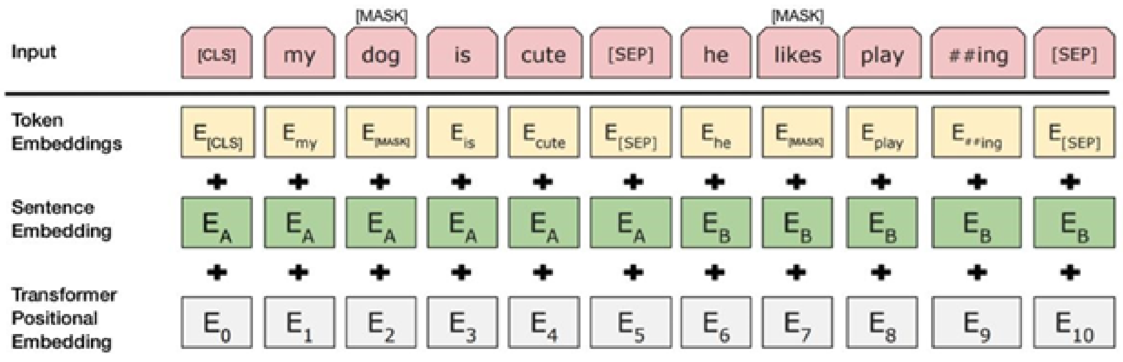
Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 5.7: BERT embedding layer

To predict if the second sentence is indeed connected to the first, the following steps are performed:

1. The entire input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 21 shaped vector, using a simple classification layer (learned matrices of weights and biases).

3. Calculating the probability of IsNextSequence with softmax.

When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

5.4.4 BERT's Architecture

BERT's model constructs with a lot of encoder layers stack over each other ,actually in case of BERT base the number of layers are 12 and in BERT large it's 24. In [24] the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. They report results on two model sizes: BERTBASE (L=12, H=768, A=12, Total Parameters=110M) and BERT LARGE (L=24, H=1024, A=16, Total Parameters=340M).

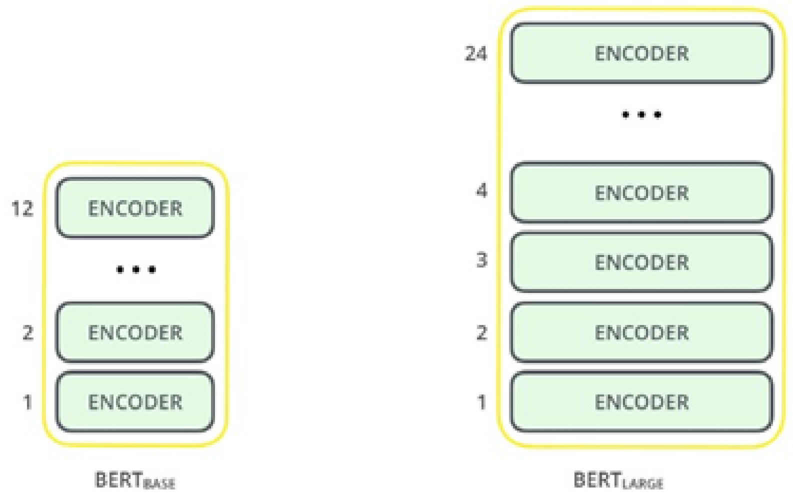

Figure 5.8: BERT base and large

BEST BASE was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left. For our implementation we use BERTBASE uncased. The model card we used, has been written by the Hugging Face team [25]

5.5 Hugging BERT Model

Hugging BERT transformers model pre-trained on a large corpus of English data in a self supervised fashion. This means it was pre-trained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pre-trained with two objectives: For Masked language modeling (MLM), taking a sentence, the model randomly masks 15the entire masked sentence through the model and has to predict the masked words. For Next sentence prediction (NSP), the models concatenate two masked sentences as inputs during pre-training. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

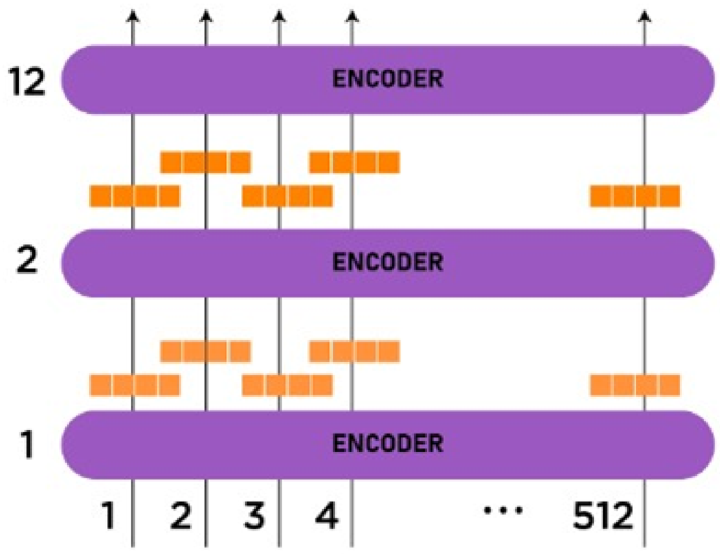

Figure 5.9: BERT base

This way, the model learns an inner representation of the English language that can then be used to extract features useful for downstream tasks: if we have a dataset of labeled sentences for instance, we can train a standard classifier using the features produced by the BERT model as inputs. But For our task we just need contextual encoded representation. So we give sentences as model input and take encoded representation as output.

5.5.1 Hugging's Model Training

Data: The Hugging's BERT model was pre-trained on Book Corpus [26], a dataset consisting of 11,038 unpublished books and English Wikipedia (excluding lists, tables and headers) [27].

Processing: The texts are lowercase and tokenized using WordPiece and a vocabulary size of 30,000. The inputs of the model are then of the form mentioned before. With probability 0.5, sentence A and sentence B correspond to two consecutive sentences in the original corpus and in the other cases, it's another random sentence in the corpus. Note that what is considered a sentence here is a consecutive span of text usually longer than a single sentence. The only constraint is that the result with the two "sentences" has a combined length of less than 512 tokens. The details of the masking procedure for each sentence are mentioned before.

Pre-Training: The model was trained on 4 cloud TPUs in Pod configuration (16 TPU chips total) for one million steps with a batch size of 256. The sequence length was limited to 128 tokens for 90used is Adam with a learning rate of 1e-4, 1 = 0.9 and 2=0.999, a weight decay of 0.01, learning rate warmup for 10,000 steps and linear decay of the learning rate after.

Chapter 6

Clustering

6.1 Introduction

Clustering is an important and powerful machine learning tool for detecting structures in labelled and unlabeled datasets. It is the procedure of dividing data objects into subclasses. In this paper we will work on unsupervised clustering, that works on datasets in which there is no outcome (target) variable nor is anything known about the relationship between the observations, that is, unlabeled data. This will give us insight into underlying patterns of different groups. Clustering quality depends on the method that is used. Clustering is also called data segmentation as large data groups are divided by their similarity. The clustering analysis that allows an object not to be part of a different cluster, or strictly belong to a single cluster is called hard clustering. On the other hand, soft clustering states that every object belongs to a cluster to a determined degree. More specific divisions can be possible to create like objects belonging to multiple clusters, to force an object to participate in only one cluster or even construct hierarchical trees on group relationships. There are different methods of clustering. named as i) Hierarchical method, ii) Partitioning method, iii) Density based method, iv) Grid-based method, v) Model-Based method [16, 17, 18]. We are going to focus on the Partitioning method.

6.2 Partitioning Clustering

Partitioning clustering decomposes a data set into a set of disjoint clusters. Say for a given data set having N points, a partitioning method constructs K partitions of the data, with each partition representing a cluster. That is, it classifies the data into K groups by satisfying the following requirements: (i) each group contains at least one point and (ii) each point belongs to exactly one group, (So (N K) this condition is must be followed (iii) The members within a same cluster are as similar as possible (iv) The members of different clusters are as dissimilar as possible. Notice that for fuzzy partitioning, a point can belong to more than one group. Partitioning clustering is done by several methods like K-means method, Expectation minimization method. In our methodology of implementation we are using the K-means method.

6.3 K-means clustering

K-means algorithm is an iterative algorithm that tries to partition the dataset into K predefined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster. Say, for a given dataset,

$$D = x_1, x_2, x_3, ..., x_n$$

K-means clustering will be applied. In K-means the intuition is, all the points of a cluster is near to the "centroid" of the cluster. Assume we have k clusters with centroids,

$$\mu_1, \mu_2, \mu_3, ..., \mu_k \in R^d$$

The term d for dimensional real vector. The way k-means algorithm works is as follows:

Specify number of clusters K. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement. Keep iterating until there is no change to the centroids, i.e. assignment of data points to clusters isn't changing. Compute the sum of the squared distance between data points and all centroids. Assign each data point to the closest cluster (centroid).Compute the centroids for the clusters by taking the average of the all data point threat belonging to each cluster.

The approach k-means follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster.

The sum of the distance of all the element of clusters

$$L = \sum_{\substack{i \;:\; i\, is\, assigned \\ to\, cluster\, j}}^{k} ||x_i - {}_j||^2 = \sum_{j=1}^{k}\sum_{i=1}^{n} a_{ij}\, ||x_i - {}_j||^2$$

Where,

$$a_i j = 1$$

if

$$x_i \; is\; assigned\; to\; j\; cluster,\; 0\, otherwise$$

. K-means tries to minimize L with respect to a and u does two steps iteratively- i) choose optimal ‘a’ for fixed centroid and ii) choose optimal centroid for fixed ‘a’. And repeats steps 1 and 2 until convergence.

$$C_j = \frac{1}{n_j}\sum x_j(a)$$

For a = 1,2,3, ... ,d.

$$C_j = means\, of\; all\, points$$

$$x_j = signed\; to\; cluster\; j\; in\, previous\, step$$

6.4 Initial Seed Selection from K-means Clustering

Clustering is one of the important unsupervised learning in data mining to group the similar features. The growing point of the cluster is known as a seed. To select the appropriate seed of a cluster is an important criterion of any seed based clustering technique. The performance of seed based algorithms are dependent on initial cluster centre selection and the optimal number of clusters in an unknown data set. Cluster quality and an optimal number of clusters are the important issues in cluster analysis. A widely used clustering algorithm, K-means in various disciplines, especially for large datasets is known to be highly sensitive to initial seed selection of cluster centres. There are two basic reasons why initial seeding or choosing initial cluster centres is important which are repeatability and reproducibility of clustering results. Generally initial cluster centres are selected randomly. K-Means does not guarantee unique clustering because we get different results with that randomly chosen initial clusters. So every different choice of initial cluster centres may lead to different clustering results which becomes a problem for Machine learning practitioners as it is difficult to rely on the results thus obtained because it is hard to know which result is better. The reason for different clustering results for K-means using different initial seeds is that

K-means algorithms try to optimize the cost function. However, a bad initial choice cannot meet the global optima rather a bad initial choice may lead it to get trapped in a local minima. Thus, with different initial choices you may get trapped in different local minima yielding different clustering results. Therefore, it is hard to repeat the clustering results. The K-means algorithm gives better results only when the initial partition is close to the nal solution (Jain and Dubes, 1988). Many studies have shown that when the initial cluster centres are close to the final cluster centres, the clustering results improve. However, it is not clear what is the right way to do so but to solve cluster centres initialization problem the following three principles are mainly followed:

1. The order of complexity of their algorithm should remain linear in the number of data objects or else it is useless because K-means is linear in the number of data objects.
2. The method should be deterministic, i.e. it should not have elements of randomness in it because on average it won't be better than random choice of initial centres.
3. The initial centre should be close enough to the final centres. This will not only make convergence faster but also help in building good clusters.

So, apparently having a non-randomization module in k-means algorithm along with a linear time complexity will provide a better result if the initial centres are closer to the final centres.

The task of finding initial cluster centres (seeding) is critical in obtaining high quality clustering for k-Means. K-Means++ (Arthur Vassilvitskii, 2007) [41] is one of the most widely used methods to solve k-Means clustering. k-means++ seeding, the state of the art algorithm, does not scale well to massive datasets as it is inherently sequential and requires k full passes through the data. The algorithm is simple and consists of two steps: In the seeding step, initial cluster centres are found using an adaptive sampling scheme called D2-sampling. In the second step, this solution is refined using Lloyd's algorithm (Lloyd, 1982), the classic iterative algorithm for k-Means. The key advantages of k-means++ are its strong empirical performance, theoretical guarantees on the solution quality, and ease of use.

6.5 Finding the value of K

The proper number of clusters that is the value of K has a direct impact on the accuracy of the system generated summary. So we have figured out that the issues that influenced the final result of our system is the selection of centroids and the value of k. There is no scope to work on the selection of centroid as it is done by random selection so the quality of the system can be developed by determining the proper value of k. For this purpose there exists many more methods like: Elbow method, Silhouette method, Gap statistic, Canopy etc. [21]. We used the Elbow method for the finding of K value.

6.5.1 The Elbow method

K-means is a simple unsupervised machine learning algorithm that groups a dataset into a user-specified number (k) of clusters. The algorithm is somewhat naïve as it clusters the data into k clusters, even if k is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters. So we can say that, the optimal number of clusters is somehow subjective and depends on the method used for measuring similarities and the parameters used for partitioning. From different methods for determining the optimal number of clusters for k-means, one method to validate the number of clusters is the Elbow method (Direct method).

Basic idea: The elbow method plots the value of the cost function produced by different values of k. If k increases, average distortion will decrease, each cluster will have fewer constituent instances, and the instances will be closer to their respective centroids. However, the improvements in average distortion will decline as k increases. The value of k at which improvement in distortion declines the most is called the elbow, at which we should stop dividing the data into further clusters.

Elbow Method Details: The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (say, k from 1 to 10), and for each value of k calculate the sum of squared errors (SSE). Then, plot a line chart of the SSE for each value of k. If the line chart resembles an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k (the SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k. Here, SSE is the sum of the squared differences between each observation and its group's mean. It can be used as a measure of variation within a cluster. If all cases within a cluster are identical the SSE would then be equal to 0. The formula for SSE is:

$$C_j = \sum_{i=1}^{n} (x_i - x^{'})^2$$

Chapter 7

Summary Extraction and Evaluation

7.1 Summary Extraction

After clustering the data, the system selects a certain number of sentences from all the clusters generated by the k-means algorithm. According to the principle of clustering sentences in k-means the elements of the same cluster (intra cluster) are the most similar to each other and most variant from the other clusters (inter cluster). So every different cluster accordingly is supposed to cover up different sub topics within the input document. To generate a summary, the human brain used to include all the subtopics of the whole document and following that the system is expected to do the same. For this purpose the system will select sentences from each cluster nearest to the centroids that is the most relevant sentence that carries the context of that cluster. But all the clusters are not meant to be the same size or frequency. So if the number of sentences picked up from each cluster is equal for all the clusters that will not expose the document in a rational way. To ensure the rational presence of information under each sub-context/ cluster we should choose more number of sentences from the cluster having higher frequency and less number of sentences from the clusters having lower frequency. To handle the naive more or less quantity, several ways can be followed. Dividing the size of all the clusters by the numerical value of the size of the smallest cluster we can get the number of sentences to be picked up from each cluster. It will be 1 for the smallest cluster and rationally greater for the others. But the ratio can be a fraction too, while a sentence can not be picked up fractionally. System will simply take the ceiling value in that case. This approach simply holds and exposes the statistical presence of sentences in the summary portraying all the subtopics and finally generates a better summary.

7.2 Summary evaluation

After developing a system it is also important to evaluate how efficiently the system works on its purpose. For the automatic text summarization system the efficiency can be measured by comparing the system generated summary with some humane made summary. There are different methods of summary evaluation such as ROUGE matrix or BLEU scores.

7.2.1 ROUGE Matric

In our experiment, for the evaluation purpose we've used the ROUGE metric. Which stands for Recall Oriented Understudy for Gisting Evaluation. It's an intrinsic metric for automatically evaluating summaries, based on BLUE, which is a metric used for machine translation. ROUGE definitely doesn't perform as good as human evaluation but as it is much more convenient that's why it has been used. In our evaluation phase we have executed different ROUGE measure: ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-L an–s ROUGE-W. Here, ROUGE-1 refers to overlap of unigrams between the system generated summary and reference summary, ROUGE-2 refers to the overlap of bigrams between the system and reference summaries and same goes for trigrams in ROUGE-3. ROUGEL measures the longest matching sequence of words using LCS (longest common sequence). An advantage of using LCS is that word order. Since it automatically includes longest in-sequence common n-grams, any predefined n-gram length is not needed. Lastly, ROUGE-W measures weighted LCS based on statistics that favours consecutive longest common sequences in between system generated summary and reference summary If we consider a document, D, the system generated summary denoted as X and reference summary as S then ROUGE is formulated as,

$$\text{ROUGE N} = \frac{\sum_{\text{S ?Reference Summary}} \sum_{\text{n grams i?S}} \min(\text{count}(i, x), \text{count}(i, s))}{\sum_{\text{S?Reference Summary}} \sum_{\text{n grams i?S}} \text{count}(i, S)}$$

The corresponding precision, recall and F1-scores are shown on the graphs. Where precision means positive predictive value that is the fraction of relevant instances (unigrams, bigrams, etc.) among the retrieved instances. In ROUGE metric precision is calculated as a ratio of number of overlapping instances (between system generated summary and reference summary) and total number of instances in system summary.

$$\text{Precision} = \frac{\text{No. of overlapping instances}}{\text{Total no. of instances in System summary}}$$

While recall, known as sensitivity which is the fraction of the total amount of relevant instances that were actually retrieved. Is calculated as the ratio of number of overlapping instances (between system generated summary and reference summary) and total number of instances in reference summary.

$$\text{Recall} = \frac{\text{No. of overlapping instances}}{\text{Total no. of instances in reference summary}}$$

Finally the F1-score is measured. The F1 score, also called F-score or F-measure, is a measure of a test's accuracy. It considers both the precision, p and the recall, r. The F1 score is the harmonic mean of the precision and recall. Formula for F-1 score:

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

F1 score can reach its best value at 1 and worst at 0.

Chapter 8

Resulting and Analysis

After building the system, through our dataset we produce their results which we called experimental results. Here we represent the results graphically and analyze them. After analyzing them we can say, we built a good system and we intend to improve more. That, we left for future work.

8.1 Experimental Result

Among the most approvable approaches for finding the K value we've used the elbow method for our experiment.

In the elbow method graph, it is not easy to know the right position of k from the graph. In that case we used python Yellowbrick library which visualizes the elbow curve and points out the position of K in the elbow curve. The following figures (fig. 8.1. to fig. 8.10.) show the elbow method and ROUGE scores graphically with corresponding documents. We use ROUGE 1, ROUGE 2, ROUGE 3, ROUGE L, ROUGE W and their recall, precision, F1 scores.

The System Summary Table below the graph provides an example of the original document of a news article and the resultant system generated summary and its reference summary. Because of the length of the report, we just include out of only one document and its generated system summary and reference summary, the maximum result could be provided on request.

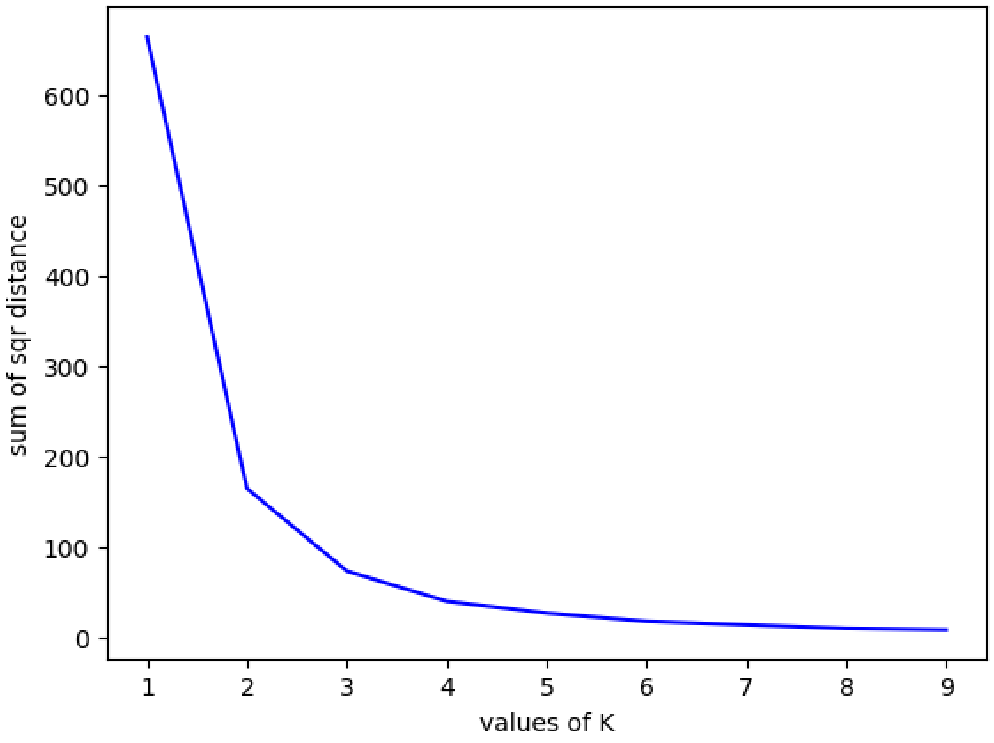

Figure 8.1: Elbow Graph for Doc1

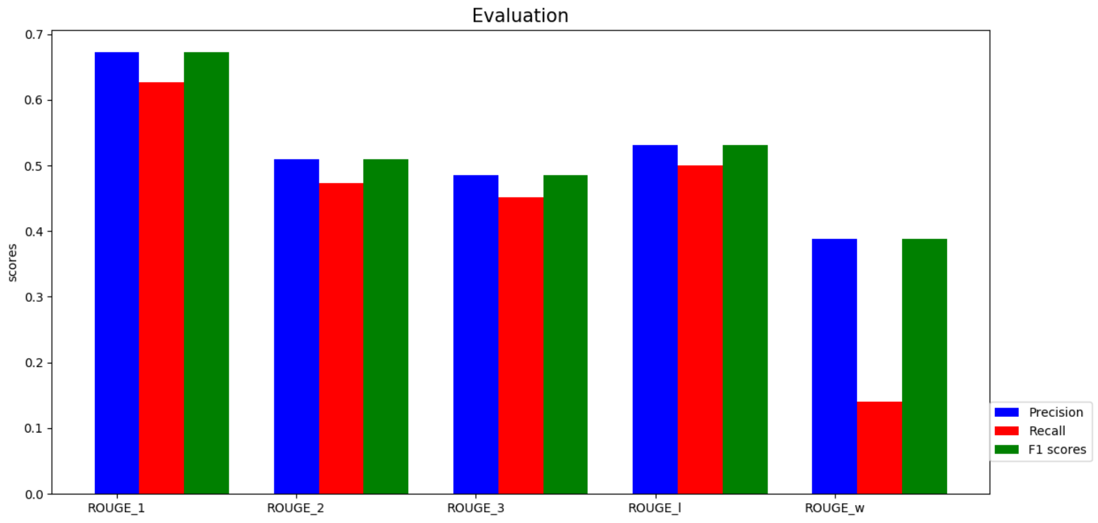

Figure 8.2: ROUGE Graph for Doc1

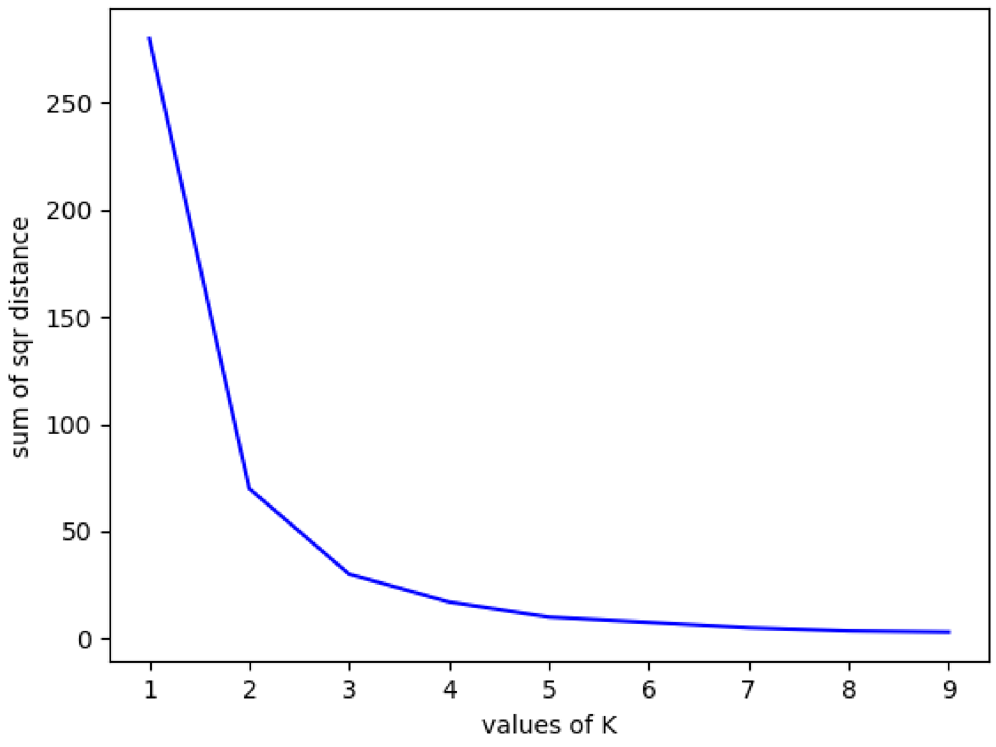

Figure 8.3: Elbow Graph for Doc2

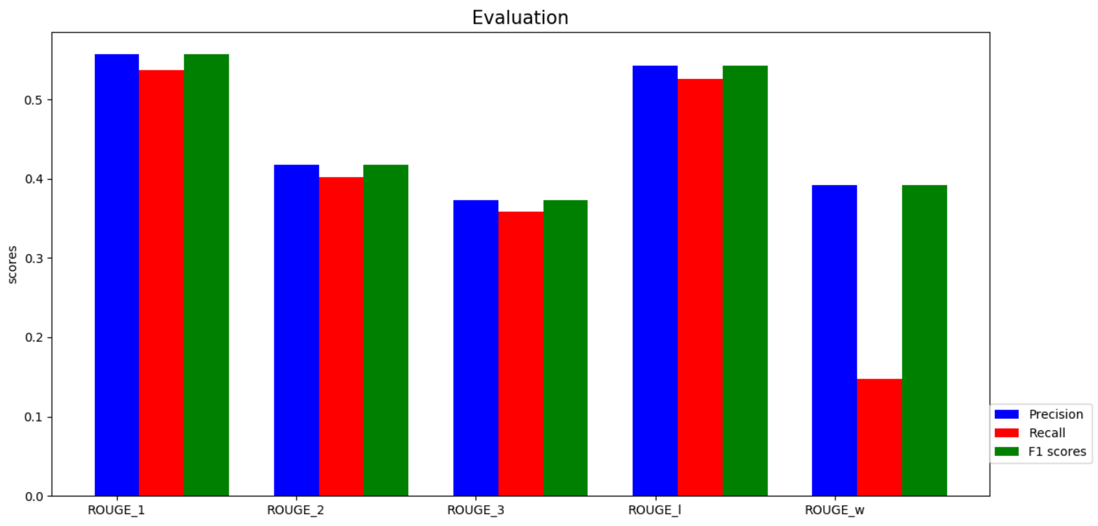

Figure 8.4: ROUGE Graph for Doc2

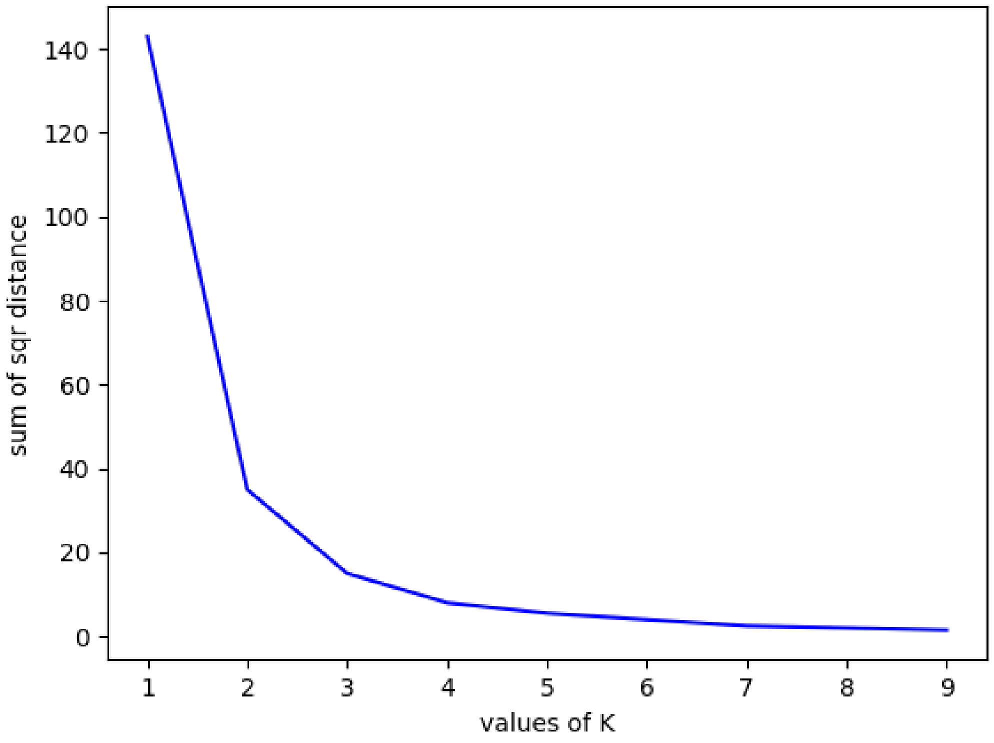

Figure 8.5: Elbow Graph for Doc3

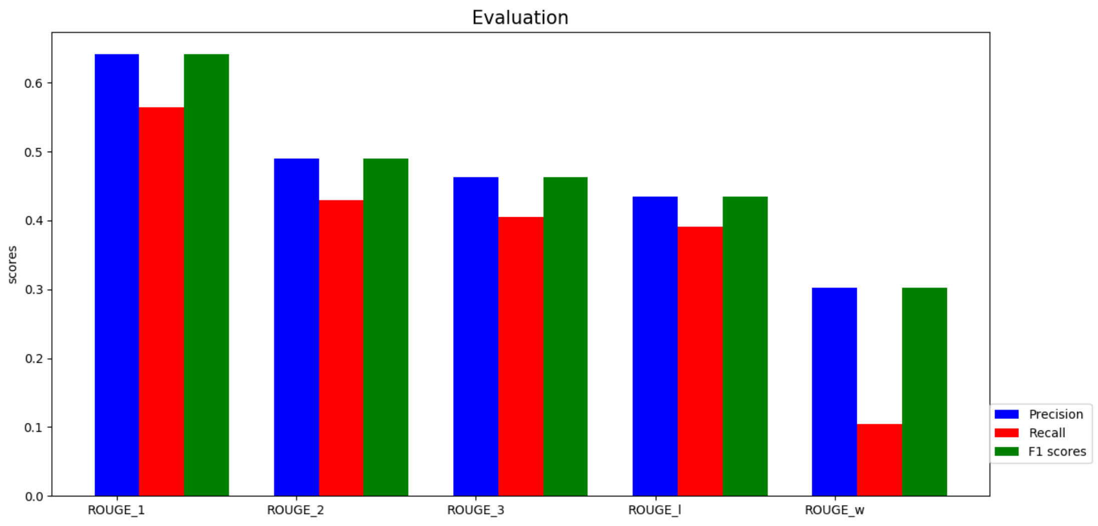

Figure 8.6: ROUGE Graph for Doc3

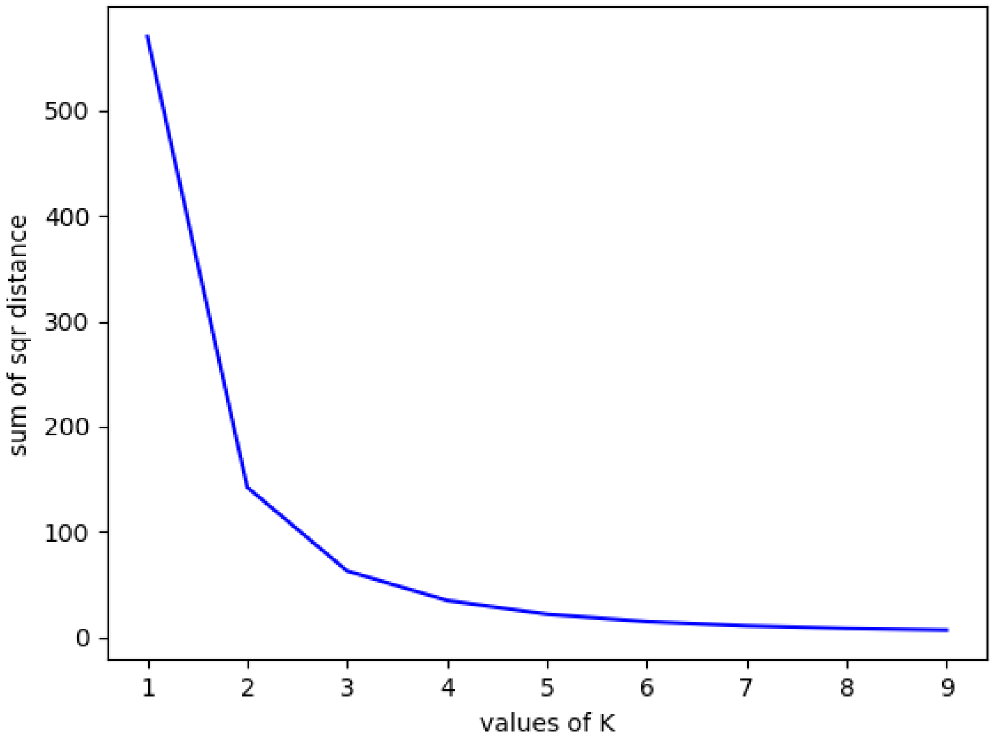

Figure 8.7: Elbow Graph for Doc4

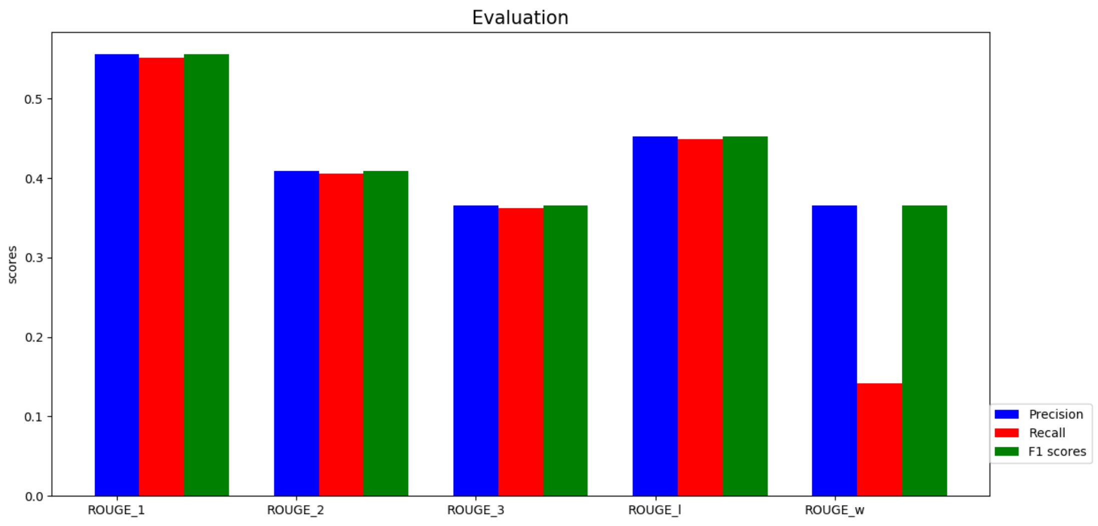

Figure 8.8: ROUGE Graph for Doc4

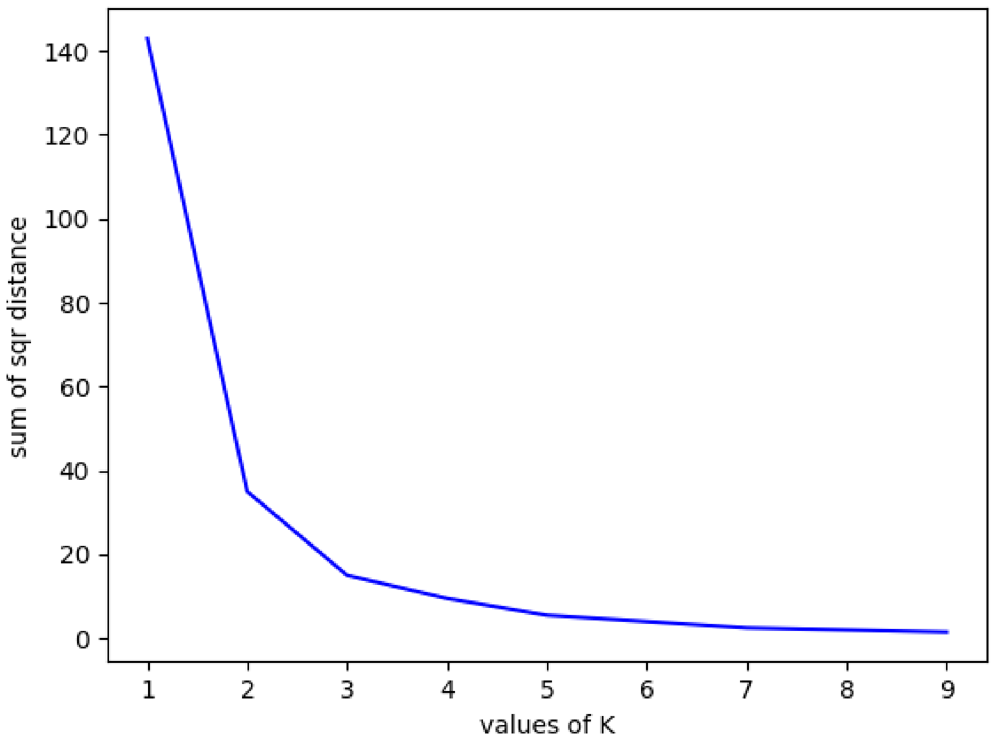

Figure 8.9: Elbow Graph for Doc5

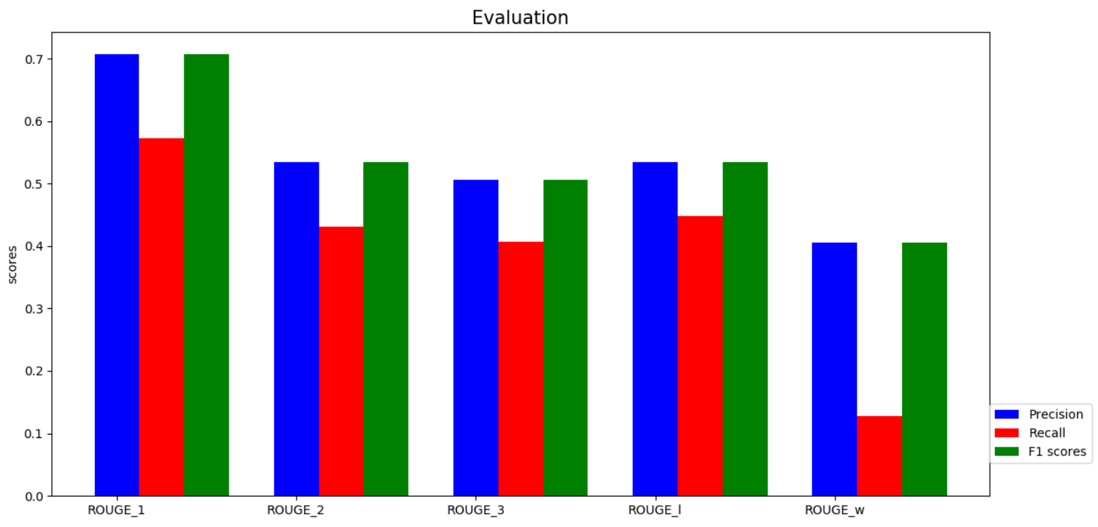

Figure 8.10: ROUGE Graph for Doc5

Example

Original News Article:

Ad sales boost Time Warner profit Quarterly profits at US media giant TimeWarner jumped 76% to $1.13bn (£600m) for the three months to December, from $639m year-earlier.The firm, which is now one of the biggest investors in Google, benefited from sales of high-speed internet connections and higher advert sales. TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn. Its profits were buoyed by one-off gains which offset a profit dip at Warner Bros, and less users for AOL.Time Warner said on Friday that it now owns 8% of search-engine Google. But its own internet business, AOL, had has mixed fortunes. It lost 464,000 subscribers in the fourth quarter profits were lower than in the preceding three quarters. However, the company said AOL's underlying profit before exceptional items rose 8% on the back of stronger internet advertising revenues. It hopes to increase subscribers by offering the online service free to TimeWarner internet customers and will try to sign up AOL's existing customers for high-speed broadband. TimeWarner also has to restate 2000 and 2003 results following a probe by the US Securities Exchange Commission (SEC), which is close to concluding.Time Warner's fourth quarter profits were slightly better than analysts' expectations. But its film division saw profits slump 27% to $284m, helped by box-office flops Alexander and Catwoman, a sharp contrast to year-earlier, when the third and final film in the Lord of the Rings trilogy boosted results. For the full-year, TimeWarner posted a profit of $3.36bn, up 27% from its 2003 performance, while revenues grew 6.4% to $42.09bn. "Our financial performance was strong, meeting or exceeding all of our full-year objectives and greatly enhancing our flexibility," chairman and chief executive Richard Parsons said. For 2005, TimeWarner is projecting operating earnings growth of around 5%, and also expects higher revenue and wider profit margins. TimeWarner is to restate its accounts as part of efforts to resolve an inquiry into AOL by US market regulators. It has already offered to pay $300m to settle charges, in a deal that is under review by the SEC. The company said it was unable to estimate the amount it needed to set aside for legal reserves, which it previously set at $500m. It intends to adjust the way it accounts for a deal with German music publisher Bertelsmann's purchase of a stake in AOL Europe, which it had reported as advertising revenue. It will now book the sale of its stake in AOL Europe as a loss on the value of that stake.

Generated Summary:

Quarterly profits at US media giant TimeWarner jumped 76% to $1.13bn (Â£600m) for the three months to December, from $639m year-earlier. The firm, which is now one of the biggest investors in Google, benefited from sales of high-speed internet connections and higher advert sales. TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn. Its profits were buoyed by one-off gains which offset a profit dip at Warner Bros, and less users for AOL. Time Warner said on Friday that it now owns 8% of search-engine Google. But its own internet business, AOL, had has mixed fortunes. It lost 464,000 subscribers in the fourth quarter profits *were lower than in the preceding three quarters.*

Reference Summary:

TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn.For the full-year, TimeWarner posted a profit of $3.36bn, up 27% from its 2003 performance, while revenues grew 6.4% to $42.09bn.Quarterly profits at US media giant TimeWarner jumped 76% to $1.13bn (£600m) for the three months to December, from $639m year-earlier. However, the company said AOL's underlying profit before exceptional items rose 8% on the back of stronger internet advertising revenues. Its profits were buoyed by one-off gains which offset a profit dip at Warner Bros, and less users for AOL. For 2005, TimeWarner is projecting operating earnings growth of around 5%, and also expects higher revenue and wider profit margins. It lost 464,000 subscribers in the fourth quarter profits were lower than in the preceding three quarters. Time Warner's fourth quarter profits were slightly better than analysts' expectations.

8.2 Experimental Analysis

The system we used during our experimental process is a computer system of CPU Intel Ⓡ Core I7 with 8 GB RAM and having Windows 10, 64 bit OS installed. The programming language is Python version 3.7 Anaconda and Spider as an IDE. Comparing the current available automatic summarizer, we can say got pretty much good results. In the experimental result section we showed ROUGE graphical representation. Now the table.2. Showed us the numeric value of ROUGE evaluation and all the ROUGE scores are in percentage. Through our dataset we got pretty impressive F1-scores. Through our experiment we mainly try to achieve three things, extracting the most important information from the original document which we can say, we achieved good enough by F1-scores, second standard summary size should be 5-35% of original document. We can say, we compressed our summary good enough by Table.3. Third, in extractive text summarizer, very often the readability is partially lost. But here in our system we have generated the summary without violating the sentence's sequences of the original document so the readability is mostly obtained.

Documents	K values using Elbow	Total number of Sentences in main Document	Total number of Sentences in Summary	Summary size
Document-1	3	20	7	35.0%
Document-2	4	28	7	25.0%
Document-3	4	29	8	27.59%
Document-4	3	15	5	33.34%

Documents	K values using Elbow	Total number of Sentences in main Document	Total number of Sentences in Summary	Summary size
Doeumcnt-5	2	8	4	50.0%

Table 8.1: Statistical Data of summaries

ROUGE	Score	Doc-1	Doc-2	Doc-3	Doc-4	Doc-5
ROUGE-1	Precision	67.29	70.00	53.92	55.45	67.96
	Recall	62.61	65.42	52.88	52.83	71.43
	F1	67.29	70.00	53.92	55.45	67.96
ROUGE-2	Precision	50.94	57.58	37.62	40.00	61.76
	Recall	47.37	53.77	36.89	38.10	64.95
	F1	50.94	57.58	37.62	40.00	61.76
ROUGE-3	Precision	48.57	56.12	35.00	34.34	59.41
	Recall	45.13	52.38	34.31	32.69	62.50
	F1	48.57	56.12	35.00	34.34	59.41
ROUGE-L	Precision	53.05	52.36	50.56	47.18	45.47
	Recall	49.95	49.49	49.75	45.31	47.39
	F1	53.05	52.36	50.56	47.18	45.47
ROUGE-W	Precision	38.87	44.78	38.58	34.10	38.83
	Recall	14.00	16.44	14.95	12.79	16.32
	F1	38.87	44.78	38.58	34.10	38.83

Table 8.2: ROUGE score for Documents

8.3 Comparison with other summarizers

To compare with other renowned summarizers, we run CNN/ Daily Mail dataset [29] on our summarizer. The CNN / Daily Mail dataset as processed by Nallapati et al. (2016) [28] has been used for evaluating summarization. The dataset contains online news articles (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average). The processed version contains 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. Models are evaluated with full-length F1-scores of ROUGE-1, ROUGE-2, and ROUGE-L.

8.3.1 Comparison Table

The following Extractive models have been evaluated on the non-anonymized version of the CNN/Daily Mail dataset introduced by [40].

Model	ROUGE-1	ROUGE-2	ROUGE-L
ImprovedBertSum(This Paper)	44.96	20.21	40.11
MatchSum [30]	44.41	20.86	40.55
DiscoBERT w.G_R & G_C [31]	43.77	20.85	40.67
eBrtSumExt [32]	43.85	20.34	39.90
BERT-ext + RL [33]	42.76	19.87	39.11
PNBERT [34]	42.69	19.60	38.85
HIBERT [35]	42.37	19.95	38.83
NeuSUM [36]	41.59	19.01	37.98
Latent [37]	41.05	18.77	37.54
BanditSum [38]	41.5	18.7	37.6
REFRESH [39]	40.0	18.2	36.6
Lead-3 baseline [40]	40.34	17.70	36.57

Chapter 9

Summary and Conclusion

9.1 Drawbacks

The drawback in extractive summarization is that it does not allow manipulating new sentences or words. But it's easier then the abstractive approach as sentences are picked up directly from the input document. A combination of these two (mostly extractive and a bit abstractive. i.e may be some connective words generated by AI) can produce a better human-like summary with less computational complication. Anyway, in our extractive text summarization system we have used BERT (base) for data featuring. BERT base models can usually feature data satisfactorily but in the case of large dataset it seems a little bit problematic. For a dataset having 200 or more sentences the system becomes slower. The k-means algorithm is quite comfortable with this amount of large dataset but the BERT base model takes a longer time to feature the dataset and also the feature will less likely to hold context for large dataset. We also used a randomly selected initial seed value for the k-means algorithm. But often the randomly selected seed leads to a local optimal while the global optima is highly expected for the better clustering.

9.2 Improvement Scope

To handle the problem with large dataset BERT (large) model can be used. It has 16 attention heads, 24 transformer layers, 340 million parameters and 1024 hidden nodes. It is so properly trained up with a huge amount of parameters. So this model is very much comfortable with any big sze dataset and features it properly. In our system we avoided the Fine Tuning phase in the BERT pretraining step. Fine tuning is done to get better results on text classification. By changing the weight for the last transformer layer in BERT better outcome for specific input dataset can be achieved. To avoid the local optima problem in k-means algorithm we can tune the hyper parameter (initial centroid) for the k-means.

9.3 Conclusion

As textual materials are enormously and rapidly extending, the necessity of systems that produce summary efficiently is growing. Text summarization refers to the process of producing a summary of a longer document which reflects the core theme of it. In our previous work we implemented a way of Extractive Text Summarization. Through the proposed algorithm we have taken an approach to make the automated summary better with the comparison to the summary generated by humans. Before executing those algorithms to summarize the main document a pre-processing step has been applied to make sure that all unnecessary characters, keywords, tags, and punctuations have been removed. Where we used TF-IDF algorithm for data featuring and K-means clustering (partitioning clustering) algorithm to generate an automated short text summary for a single document or a set of documents picking the most substantial sentences. But before proceeding to any kind of grouping/-clustering/similarity check the features should be more contextfull. For better understanding of human language finally we have used the BERT model of Transformer to feature the dataset that pays more attention to the context of the document. Hence the summary is more meaningful. Automated summarization is an important field of research in NLP (Natural Language Processing) research. It consists of automatically creating a summary of a single or multiple text document. The purpose of extractive document summarization is to automatically select a number of indicative sentences, passages, or paragraphs from the original document Most summarization techniques are based on extractive methods. An Abstracted method is similar to summaries made by humans.

References

1. Christian Guldena, Melanie Kirchnerb, Christina Schüttlera, Marc Hinderera, Marvin Kampfb, Hans-Ulrich Prokoscha, Dennis Toddenrotha “Extractive summarization of clinical trial descriptions”-21 May 2019.

2. Rahim Khan, Yurong Qian, Sajid Naeem “Extractive based Text Summarization Using KMeans and TF-IDF”-3 May 2019.

3. Sean MacAvaney, Nazli Goharian, Ish Talati, “Ontology-Aware Clinical Abstractive Summarization”-14 May 2019.

4. Feng Li, Yan Chen, Zhoujun Li “Learning from the past: Improving news summarization with past news articles”- 25 Oct. 2015

5. Changjian Fang, Dejun Mu, Zhenghong Deng, Zhiang Wu “Word-sentence coranking for automatic extractive text summarization”-5 March 2016.

6. René Arnulfo García-Hernández, Yulia Ledeneva “Word Sequence Models for Single Text Summarization” 2009 Second International Conferences on Advances in Computer-Human Interactions.

7. Mohammad Reza Keyvanpour, Mehrnoush Barani Shirzad, Haniyeh Rashidghalam “ELTS: A Brief Review for Extractive Learning-Based Text Summarizaton Algorithms” 5th International Conference on web research.

8. Ledeneva Y., Gelbukh A., García-Hernández R.A. (2008) Terms Derived from Frequent Sequences for Extractive Text Summarization. In: Gelbukh A. (eds) Computational Linguistics and Intelligent Text Processing. CICLing 2008. Lecture Notes in Computer Science, vol 4919. Springer, Berlin, Heidelberg

9. Yulia Ledeneva, Alexander Gelbukh, and René Arnulfo García-Hernández “Extractive Text Summarization Using Sentence Ranking”- 2019 International Conference on Data Science and Communication (IconDSC)

10. Xiangke Mao,Hui Yang,Shaobin Huang,Ye Liu,Rongsheng Li "Extractive summarization using supervised and unsupervised learning" Expert Systems with Applications -November 2019.

11. Ferreira, Rafael, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George DC Cavalcanti, Rinaldo Lima, Steven J. Simske, and Luciano Favaro. "Assessing sentence scoring techniques for extractive text summarization." Expert systems with applications 40, no. 14 (2013): 57555764.

12. Yohei SEKI 2003, “Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles” National Institute of Informatics.

13. Lloret, E., 2008. Text summarization: an overview. Paper supported by the Spanish Government under the project TEXT-MESS (TIN2006-15265-C06-01).

14. www.kaggle.com/datasets

15. Y,S,Patail , M.B. Vaidya 2012, “A Technical survey on Clustering Analysis in Data mining” International Journal of Emerging Technology and Advanced Engineering.

16. Himanshu Gupta, Dr.Rajeev Srivastav 2014, “K-means Based Document Clustering with Automatic ‘K’ Selection and Cluster Refinement” International Journal of Computer Science and Mobile Applications.

17. Greg Hamerly and Charles Elkan 2003, “Learning the k in k- means” In Neural Information Processing System, MIT Press.

18. Chun-ling Chen,S.C. Tseng and Tyne Liang Nov. 2010, “An integration of Word Net and Fuzzy association rule mining for multi-label document clustering” Data and Knowledge Engineering, pp. 1208-1226.

19. Mehdi Allahyari.et al. August 2017, “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques” In Proceedings of KDD Bigdas, Halifax, Canada, 13 pages.

20. Neepa Shah, Sunita Mahajan October 2012, “Document Clustering: A Detailed Review” International Journal of Applied Information Systems (IJAIS) Vol. 4.

21. Nazeer, K.A. and Sebastian, M.P., 2009, July. Improving the Accuracy and Efficiency of the k-means Clustering Algorithm. In Proceedings of the world congress on engineering (Vol. 1, pp. 1-3). London: Association of Engineers.

22. Gupta, V. and Lehal, G.S., 2010. A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence, 2(3), pp.258268.

23. https://yashuseth.blog/2019/06/12/bert-explained-faqs-understand-bertworking/: :text=What

24. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pretraining of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

25. https://huggingface.co/bert-base-uncased

26. https://yknzhu.wixsite.com/mbweb

27. https://en.wikipedia.org/wiki/EnglishWikipedia

28. Nallapati, Ramesh, et al. "Abstractive text summarization using sequence-tosequence rnns and beyond." arXiv preprint arXiv:1602.06023 (2016).

29. Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P. (2015). Teaching machines to read and comprehend. In Advances in neural information processing systems (pp. 1693-1701).

30. Zhong M, Liu P, Chen Y, Wang D, Qiu X, Huang X. Extractive Summarization as Text Matching. arXiv preprint arXiv:2004.08795. 2020 Apr 19.

31. Xu J, Gan Z, Cheng Y, Liu J. Discourse-aware neural extractive model for text summarization. arXiv preprint arXiv:1910.14142. 2019 Oct 30.

32. Liu, Yang, and Mirella Lapata. "Text summarization with pretrained encoders." arXiv preprint arXiv:1908.08345 (2019).

33. Bae S, Kim T, Kim J, Lee SG. Summary level training of sentence rewriting for abstractive summarization. arXiv preprint arXiv:1909.08752. 2019 Sep 19.

34. Zhong M, Liu P, Wang D, Qiu X, Huang X. Searching for Effective Neural Extractive Summarization: What Works and What's Next. arXiv preprint arXiv:1907.03491. 2019 Jul 8.

35. Zhang X, Wei F, Zhou M. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. arXiv preprint arXiv:1905.06566. 2019 May 16.

36. Zhou Q, Yang N, Wei F, Huang S, Zhou M, Zhao T. Neural document summarization by jointly learning to score and select sentences. arXiv preprint arXiv:1807.02305. 2018 Jul 6.

37. Zhang, X., Lapata, M., Wei, F. and Zhou, M., 2018. Neural latent extractive document summarization. arXiv preprint arXiv:1808.07187.

38. Dong, Yue, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. "Banditsum: Extractive summarization as a contextual bandit." arXiv preprint arXiv:1809.09672 (2018).

39. Narayan, Shashi, Shay B. Cohen, and Mirella Lapata. "Ranking sentences for extractive summarization with reinforcement learning." arXiv preprint arXiv:1802.08636 (2018).

40. See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017)