DISSERTATION

ON

TEXT SEGMENTATION USING BERT FOR TEXT SUMMARIZATION

Thesis Submitted in the partial fulfilment of the requirements for the degree of

Master of Engineering

In

Computer Science & Engineering

Submitted By

Rahul Burman

Examination Roll number: M4CSE22027

Registration number:154151 of 2020-2022

Under Guidance of

Prof. (Dr.) Kamal Sarkar

Jadavpur University

Dept. of Computer Science & Engineering Faculty Council of Engineering and Technology JADAVPUR UNIVERSITY

KOLKATA – 700032

2020 – 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that Rahul Burman (Examination Roll number:M4CSE22027) has completed her dissertation entitled "Text Segmentation using BERT for Text Summarization", under the supervision and guidance of Prof. (Dr.) Kamal Sarkar, Jadavpur University, Kolkata. We are satisfied with his work, which is being presented for the partial fulfilment of the degree of Master of Engineering in Computer Science & Engineering, Jadavpur University, Kolkata - 700032.

…………………………………………

Prof. Anupam Sinha

Head of the Department

Department of Computer Science & Engineering

Jadavpur University, Kolkata - 700032

………………………………….

Prof. Kamal Sarkar,

Project Supervisor

Department of Computer Science & Engineering

……………………………………

Prof. Chandan Mazumdar

Dean , Faculty council of Engg. & Tech.

Jadavpur University, Kol - 700032

JADAVPUR UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

Certificate of Approval*

The foregoing thesis entitled "Text Segmentation using BERT for Text Summarization," is hereby approved as a creditable study of Master of Engineering in Computer Science & Engineering and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion therein but approve this thesis only for the purpose for which it is submitted.

FINAL EXAMINATION FOR

EVALUATION OF PROJECT

1. ______________________________

2. ________________________________

Signature of Examiners

* *Only in case the thesis is approved*

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY - 700032

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her Master of Engineering in Computer Science & Engineering.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name (in block): RAHUL BURMAN

University Roll No. 002010502027

Examination Roll Number: M4CSE22027

Registration Number: 154151 of 2020-2022

Thesis Title: "Text Segmentation using BERT for Text Summarization "

Signature with date:

Acknowledgement

I would like to express my honest and sincere thanks and humble gratitude to my honourable teacher and guide Prof. Kamal Sarkar, Professor of the Department of Computer Science & Engineering, Jadavpur University, for his exclusive guidance and full support in completing and producing this project successfully. I am very much obliged to him for the constant encouragement and continuous inspiration that he has given to me. These words are only a token of my deep respect towards him for all he has done to take my project to the present shape.

I would like to thank Mrs. Sohini Roychowdhuri for all her support and valuable suggestions to the activities of the project at any phase of the project.

……………………..

Rahul Burman

Examination Roll No - M4CSE22027

Registration No - 154151

TABLE OF CONTENTS

Abstract

Text summarization creates an automatically generated summary that includes all pertinent significant information from the original material. When looking at the summary results, extractive and abstractive techniques are one of the most common. In the near future, an extracted summary will be available. Extractive summarization selects a subset of sentences from the text to form a summary. In this work we have used the BERT model for Text Summarization using the method of Text Segmentation to achieve our goal.

1. Introduction

People have become overwhelmed by the vast amount of information and documents available on the internet as the internet and big data have grown in popularity. Many researchers are motivated to develop a technology solution that can automatically summarise texts as a result of this. Text summarization is the technique for generating a concise and precise summary of voluminous texts while focusing on the sections that convey useful information, and without losing the overall meaning. Automatic text summarization aims to transform lengthy documents into shortened versions, something which could be difficult and costly to undertake if done manually. Machine learning algorithms can be trained to comprehend documents and identify the sections that convey important facts and information before producing the required summarised texts.. As a result, the information arrives efficiently while maintaining the document's original objective.Text summarization research has been explored since the mid-twentieth century, with Lun in 1958 being the first to openly address it using a statistical technique called word frequency diagrams. To date, many different ways have been developed.There are single and multi-document summary options based on the number of documents. Meanwhile, the extractive and abstractive outcomes are based on the summary results. A single document generates a summary from a single source document, and the content described is all about the same thing. Nevertheless, the multi-document summarization is based on a variety of sources or papers that talk about the same thing. There are mainly four types of summaries. They are 1. Single document Summary, 2. Multi-document summary, 3. Query focused Summary, 4. Informative Summary.

- Single document summary basically summarises from a single source of document and gives the important information.
- Multi - document summary basically summarises from multiple sources of documents and gives the important information.
- Query focused summary basically summarises from a specific query and provides the important information.
- Informative summary : It includes a summary of the full information.

There are two approaches for summarization.

1. **Extractive-based Summarization** : Extracting the most relevant phrases and lines from the document is part of the extractive approach. The summary is then created by combining all of the essential lines. As a result, every line and word of the summary belongs to the original document that is summarised in this situation.
2. **Abstractive- based Summarization** : Summarization based on deep learning is used in the abstractive approach. As a result, it employs new phrases and terms that differ from those found in the original document while maintaining the same main themes,

much like we do when we summarise. As a result, it is significantly more difficult than the extractive method.

Application of Text Summarization is as follows:

1. Newsletters : Many weekly newsletter begin with an introduction and then feature a carefully curated selection of relevant content. Summarization would allow businesses to supplement newsletters with a stream of summaries (rather than a list of links), which is a more mobile-friendly format.
2. Financial Research : Investment banking firms spend a lot of money on large data to help them make decisions, which includes computerised stock trading. When you work as a financial analyst and read market reports and news on a daily basis, you will eventually reach a point when you can no longer read everything. Financial document summarization systems, such as earnings reports and financial news, can aid analysts in quickly extracting market signals from information.
3. Social media marketing : Long-form content producers, such as whitepapers, e-books, and blogs, may be able to use summarising to break down their information and make it shareable on social media sites like Twitter and Facebook. Companies would be able to reuse current content more effectively as a result of this.
4. SEO and Search Marketing: It's vital to have a thorough understanding of what your competitors are talking about in their content when evaluating search queries for SEO. This is especially crucial now that Google has modified its algorithm and moved its focus to topical authority. Multi-document summary is a useful tool for swiftly analysing a large number of search results, identifying common themes, and skimming the most significant aspects.
5. Internal Document Workflow : Internal knowledge is regularly produced by large firms, and it is usually kept and underutilised in databases as unstructured data. These businesses should embrace solutions that allow them to reuse previously acquired information. Analysts can use summarization to swiftly grasp what the company has done in a specific subject and quickly put together reports that include multiple points of view.
6. Question answering and Bots : The smart home and the office are being taken over by personal assistants. Most assistants, on the other hand, are constrained to relatively particular jobs. Summarization on a large scale could be a useful strategy for answering questions. A summarizer could create a cohesive answer in the form of a multi-document summary by collecting the most relevant papers for a certain inquiry.
7. Video Scripting : One of the most significant marketing mediums is video. People are now uploading videos on professional networks like LinkedIn, in addition to video-focused websites like YouTube or Vimeo. Scripting may or may not be required, depending on the type of video. When it comes to writing a script that integrates research from a variety of sources, summarization can be a valuable ally.
8. Medical Cases : With the rise of telehealth, there is a rising need to effectively handle medical situations that are now completely digital. Telemedicine networks promise a more open and accessible healthcare system, but the procedure must be scalable.

When it comes to assessing medical situations and directing them to the proper health expert, summarization can be a critical component in the tele-health supply chain.

9. Video-conferencing and meeting : The capacity to record crucial ideas and content from discussions is becoming increasingly important as teleworking becomes more popular. It would be amazing if there was a technology that could convert audio to text and provide summaries from your team meetings.
10. Customer service and help desk : Knowledge bases have been around for a while, and they're essential for SAAS platforms to enable scaled client assistance. Even yet, when reading assistance documentation, users may feel overwhelmed. Multi-document summary brings together key elements from a variety of assistance articles to provide the user a complete picture of the problem.
11. Social media marketing : Long-form content producers, such as whitepapers, e-books, and blogs, may be able to use summarising to break down their information and make it shareable on social media sites like Twitter and Facebook. Companies would be able to reuse current content more effectively as a result of this.
12. Science and Research & Development : A human-written abstract that serves as a summary is common in academic articles. When you're responsible with tracking trends and innovation in a certain industry, though, reading every abstract might get daunting. This work may benefit from systems that can group papers and condense abstracts further.
13. Patent Research : Patent research can be a time-consuming procedure. Whether you're conducting market research or preparing to file a new patent, using a summarizer to extract the most important claims from multiple patents could save you time.
14. Assistance to disabled people : People with hearing difficulties may benefit from summarising as voice-to-text technology improves, allowing them to keep up with content more efficiently.

In this thesis, I have applied the bert model for text summarization using text segmentation. The text segmentation is done with a certain condition that should fulfil and is mentioned in detail in the methodology part.

2. Literature Survey

There have been many works done in the field of Text Summarization using different approaches and methods using varieties of algorithms. In this section we will see some past works done in Text Summarization.

2.1 Topic Representation Approaches

In this section we describe some of the most widely used topic representation approaches.

2.1.1 Topic words

One of the most prominent topic representation approaches is the topic words technique, which seeks to find words that characterise the input document's topic. [38] was one of the first works to use this method, which used frequency thresholds to locate descriptive words in a document and represent the document's topic. [22] offered a more advanced version of Luhn's approach, in which they employed the log-likelihood ratio test to find explanatory phrases, which are referred to as the "theme signature" in summarization literature. In the news sector, using topic signature terms as topic representation was quite effective and improved multi-document summarization accuracy [29]. See [46] for further details on the log-likelihood ratio test.

The relevance of a sentence can be calculated in two ways: as a function of the number of subject signatures in the sentence or as a proportion of the topic signatures in the sentence. Both sentence scoring functions use the same topic representation, but they may give sentences different scores. Longer sentences may receive higher values under the first method since they contain more words. The density of the topic terms is measured using the second method.

2.1.2 Frequency driven approach

We can use binary (0 or 1) or real-value (continuous) weights to determine which words are more connected to the topic when assigning weights to terms in topic representations. Word probability and TFIDF are the two most used strategies in this category (Term Frequency Inverse Document Frequency).

2.1.2.1 *Word probability* : Word probability is the simplest way for using frequency of words as indicators of importance. The number of occurrences of a word, f (w), divided by the total number of words in the input (which can be a single document or numerous documents) yields the likelihood of the word w:

$$P(w) = \frac{f(w)}{N} \qquad (1)$$

Vanderwende et al. [75] proposed the SumBasic system which uses only the word probability approach to determine sentence importance. For each sentence, S_j , in the input, it assigns a weight equal to the average probability of the words in the sentence.

$$g(S_j) = \frac{\sum_{w_i \in S_j} P(w_i)}{|\{w_i \mid w_i \in S_j\}} \qquad (2)$$

Where $g(S_j)$ is the weight of sentence S_j.

It then chooses the best-scoring sentence with the highest probability word as the next phase. This stage ensures that the summary contains the highest probability word that represents the document's topic at the time. The weight is then updated for each word in the chosen sentence:

$$p_{new}(w_i) = p_{old}(w_i)p_{old}(w_i) \qquad (3)$$

This modification to the wordweight indicates that the likelihood of a word appearing in the summary is lower than the likelihood of a word appearing only once.
The preceding selection processes will be repeated until the required summary length is obtained. SumBasic's sentence selection strategy is based on the greedy strategy. Yih et al. [79] employed an optimization strategy (as a sentence selection strategy) to increase the occurrence of important words throughout the summary. Another example of applying an optimization strategy is [2.]

2.1.2.2 *TFIDF.* There is a need for more advanced techniques because word probability strategies rely on a stop word list to exclude them from the summary, and determining which words to include in the stop list is not easy.
TFIDF is one of the more complex and widely used methods for assigning weight to words (Term Frequency Inverse Document Frequency).
By assigning low weights to words that exist in most texts, this weighing technique examines the relevance of words and detects very common terms (that should be excluded from consideration) in the document(s). The following formula is used to calculate the weight of each word w in document d:

$$q(w) = f_d(w) * log\frac{|D|}{f_D(w)} \qquad (4)$$

where $f_d(w)$ is term frequency of word w in the document d, $f_D(w)$ is the number of documents that contain word w and |D| is the number of documents in the collection D. For more information about TFIDF and other term weighting schemes, see [59]. TFIDF weights are easy and fast to compute and also are good measures for determining the importance of sentences, therefore many existing summarizers [2, 3, 24] have utilised this technique(or some form of it).

TFIDF topic representation is used in centroid-based summarization, another set of approaches that has become a standard baseline. This method uses a set of features to compute the salience of sentences and rank them accordingly. [55] provides a comprehensive discussion of the centroid-based technique, but we'll just go over the basics here.

The first step is topic detection and documents that describe the same topic clustered together. To achieve this goal, TFIDF vector representations of the documents are created and those words whose TF IDF scores are below a threshold are removed. Then, a clustering algorithm is run over the TF IDF vectors, consecutively adding documents to clusters and recomputing the centroids according to:

$$c_j = \frac{\sum_{d \in C_j} d}{|C_j|} \qquad (5)$$

where cj is the centroid of the jth cluster andCj is the set of documents that belong to that cluster. Centroids can be considered as pseudo- documents that consist of those words whose TF IDF scores are higher than the threshold and form the cluster.

The next step is to use centroids to find sentences in each cluster that are key to the cluster's overall theme. Two metrics are defined to achieve this goal [54]: cluster-based relative utility (CBRU) and cross-sentence informative subsumption (CSIS). CBRU determines how significant a sentence is to the overall topic of the cluster, while CSIS assesses sentence redundancy. Three features (central value, positional value, and first-sentence overlap) are used to approximate two measures. Next, the final score of each sentence is computed and the selection of sentences is determined. For another related work, see [76].

2.1.3 Latent Semantic Analysis

Latent semantic analysis (LSA) which is introduced by [20], is an unsupervised method for extracting a representation of text semantics based on observed words. Gong and Liu [25] initially proposed a method using LSA to select highly ranked sentences for single and multi-document summarization in the news domain. The LSA method first builds a term-sentence matrix (n by m matrix), where each row corresponds to a word from the input (n words) and each column corresponds to a sentence (m sentences). Each entry ai j of the matrix is the weight of the word i in sentence j. The weights of the words are computed by TFIDF technique and if a sentence does not have a word the weight of that word in the sentence is
zero. Then singular value decomposition (SVD) is used on the matrix and transforms the matrix A into three matrices: $A = U\Sigma V^T$.

MatrixU (n × m) represents a term-topic matrix having weights of words. Matrix is a diagonal matrix (m × m) where each row i corresponds to the weight of a topic i. Matrix VT is the topic sentence matrix. The matrix $D = \sum V^T$ describes how much a sentence represents a topic, thus, d_{ij} shows the weight of the topic i in sentence j.

Gong and Liu's strategy was to choose one sentence for each topic, therefore they kept the number of topics dependent on the length of the summary in terms of sentences. Because a topic may require more than one sentence to convey its information, this technique has a disadvantage. As a result, different ways for improving the performance of LSA-based summarising algorithms have been presented. One improvement was to use the weight of each topic to determine the relative size of the summary that should cover the issue, allowing for a variable number of sentences to be used. [68] describes yet another advancement. Steinberger et al. [68] proposed an LSA-based method that outperforms the original work by a significant margin. They noticed that sentences that discuss some of the most essential issues are suitable candidates for summaries, thus they defined the weight of the sentence as follows to locate those sentences:

Let g be the weight function, then

$$g(s_i) = \sqrt{\sum_{j=1}^{m} d_{ij}^2} \qquad (6)$$

For other variations of LSA technique, see [27, 50].

2.1.4 Bayesian Topic Models

Many existing multi-document summarization approaches suffer from two drawbacks [77]: 1) Because they believe sentences to be independent of one another, they ignore subjects that are embedded in the documents. 2) Most present systems do not compute sentence scores with particularly clear probabilistic meanings, and many of the sentence scores are calculated using heuristics.

Bayesian topic models are probabilistic models that uncover and represent the topics of documents. They are quite powerful and appealing, because they represent the information (i.e. topics) that are lost in other approaches. Their advantage in describing and representing topics in detail enables the development of summarizer
systems which can determine the similarities and differences between documents to be used in summarization [39].

Topic models frequently include a separate method for rating the sentence called Kullbak-Liebler, in addition to improving topic and document representation (KL). The KL [34] is a difference (divergence) metric between two probability distributions P and Q. The KL divergence of Q from P over the words w is defined as: In summarization, where we have probability of words, the KL divergence of Q from P over the words w is defined as:

$$D_{KL}(P||Q) = \sum_{w} P(w) log \frac{P(w)}{Q(w)} \qquad (7)$$

Where P(w) and Q(w) are probabilities of w in P and Q.

KL divergence is an interesting method for scoring sentences in the summarization, because it shows the fact that good summaries are intuitively similar to the input documents. It describes how the importance of words alters in the summary in comparison with the input, i.e. the KL divergence of a good summary and the input will be low.

In recent years, probabilistic topic models have gotten a lot of interest in a variety of fields [4–7, 17, 28, 44, 57]. The state-of-the-art unsupervised technique for extracting thematic information (themes) from a collection of documents is the Latent Dirichlet allocation (LDA) model. [12, 69] provides a comprehensive overview of LDA, however the basic principle is that documents are represented as a random mixture of latent themes, each of which is a probability distribution over words.

Recently, LDA has been widely employed for multi-document summarization. For a query-focused summary, Daume et al. [19] presented BayeSum, a Bayesian summarization model. Wang et al. [77] proposed a Bayesian sentence-based topic model for summarization that took into account both term-document and term- sentence relationships. Their methodology exceeded several other summarising methods in terms of significance. Multi-document summarization is described by Celikyilmaz et al. [13] as a prediction issue based on a two-phase hybrid model. To begin, they offer a hierarchical topic model for determining all sentences' topic structures. After that, they use an unique tree-based sentence scoring mechanism to compare candidate sentences to human-provided summaries. They utilise these scores in the second stage to train a regression model based on the lexical and structural properties of the phrases, and then use the model to score sentences from new documents (unseen documents) to create a summary.

2.2 KNOWLEDGE BASES AND AUTOMATIC SUMMARIZATION

The goal of artificial text summarising is to produce summaries that look and feel like summaries generated by humans. However, the soundness and readability of produced summaries are frequently unsatisfactory, since the summaries fail to adequately include all semantically relevant parts of material. This is due to the fact that many existing text summarising approaches ignore word semantics. Combining summary techniques with knowledge bases (semantic-based or ontology-based summarizers) is one step toward developing more accurate summarization systems.
Human-generated knowledge bases and diverse ontologies in a variety of fields (e.g., Wikipedia, YAGO, DBpedia, etc.) have opened up new possibilities in text summarization, and have recently gotten a lot of attention. Henning et al. [30], for example, describe a method for language extraction that maps sentences to ontology concepts. They can improve the semantic representation of phrases by considering ontology aspects, which is useful for selecting sentences for summaries. They demonstrated that ontology-based sentence extraction outperforms baseline summarizers in an experiment. Chen et al. [16] provide a user query-based text summarizer that creates a summary for medical material using the UMLS medical ontology. Baralis et al. [10] offer a Yago-based summarizer that makes use of the YAGO ontology [70] to find essential topics in documents. The concepts are assessed, and the most representative document sentences are chosen. Sankarasubramaniam et al. [60] present a method. Techniques for Text Summarization. A Brief Survey arXiv, July 2017, USA that employs Wikipedia in conjunction with a graph-based ranking technique. First, they create a bipartite sentence-concept graph, and then use an iterative ranking algorithm for selecting summary sentences.

2.3 INDICATOR REPRESENTATION APPROACHES

Rather than describing the themes of the input text, indicator representation approaches try to model the representation of the text based on a collection of attributes and use them to directly rank the sentences. To identify which sentences should be included in the summary, graph-based methods and machine learning techniques are frequently used.

2.3.1 Graph method for summarization

The documents are represented as a connected network by graph methods, which are influenced by the PageRank algorithm [42]. The vertices of the graph are sentences, and the edges between them reflect how similar the two sentences are. A typical method for connecting two vertices is to compare the similarity of two words and connect them if the similarity is greater than a certain threshold. Cosine similarity with TFIDF weights for words is the most commonly used approach for determining similarity.

There are two outputs from this graph representation. To begin, the graph's partitions (sub-graphs) create separate subjects covered in the papers. The identification of the document's essential sentences is the second output. Sentences that are connected to a large number of other sentences in the partition are likely to be at the centre of the graph and included in the summary.

Graph-based approaches can be used to summarise single or several documents [24]. They can be used to a variety of languages because they do not require language-specific linguistic processing other than sentence and word boundary recognition [43].However, employing the TF IDF weighting technique for similarity measurement has drawbacks because it only preserves word frequency and ignores syntactic and semantic information. As a result, similarity metrics based on syntactic and semantic information improve the summarization system's performance [14]. See [46] for more graph-based techniques.

2.3.2 Machine learning for Summarization

Summarization is modelled as a classification problem in machine learning methodologies. [35] is a preliminary study that attempts to use machine learning approaches for summarization. Given a training set of texts and their extracted summaries, Kupiec et al. design a classification function, the naive-Bayes classifier, to categorise sentences as summary sentences or non-summary sentences based on the features they have. Using Bayes' rule, the classification probabilities are learned statistically from the training data:

$$P(s \in S|F_1, F_2, ..., F_k) = \frac{P((F_{1,} F_2, ..., F_k)|s\in S)P(s\in S)}{P(F_1, F_2, ..., F_k)} \qquad (8)$$

where s is a sentence from the document collection, F_1, F_2, . . . , F_k are features used in classification and S is the summary to be generated. Assuming the conditional independence between the features:

$$P(s \in S|F_1, F_2, \ldots\ldots, F_k) = \frac{\prod_{i=1}^{k} P((F_i)|s\in S)P(s\in S)}{\prod_{i=1}^{k} P(F_i)} \tag{9}$$

The score of a sentence determines its likelihood of being included in the summary. The classifier you choose serves as a sentence scoring mechanism. The position of sentences in the document, sentence length, presence of uppercase words, resemblance of the sentence to the document title, and so on are all common features employed in summary. [48, 78, 80], to name a few, have applied machine learning algorithms in summarization.

Machine learning algorithms for summarization include Naive Bayes, decision trees, support vector machines, Hidden Markovmodels, and Conditional RandomFields. One key distinction between classifiers is that the inclusion of phrases in the summary must be selected independently. It turns clear that methods like the Hidden Markov model [18] and Conditional Random Fields [65] that explicitly assume sentence dependency outperform other strategies.

One of the main drawbacks of using supervised learning methods for summarising is that they require a collection of training documents (labelled data), which are not always readily available. Alternatives to dealing with this problem have been presented by researchers:

- **Annotated corpora creation** : The researchers will profit immensely from the creation of an annotated corpus for summarization because more public benchmarks will be available, making it easier to evaluate different summarization algorithms. With insufficient data, it also reduces the risk of overfitting. Ulrich et al. [74] describe the process of creating a publicly available annotated email corpus. However, constructing an annotated corpus takes a long time, and there is no uniform agreement on which sentences to use, so various persons may use different sentences to construct the summary.
- **Semi - supervised approaches** : To train a classifier, a semi-supervised approach is used. In semi-supervised learning, unlabeled data is used in training. A little amount of labelled data is generally present alongside a huge amount of unlabeled data. [15] provides a comprehensive overview of semi-supervised learning. For extractive summarization, Wong et al. [78] presented a semi-supervised technique.They iteratively trained two classifiers to exploit unlabeled data. The unlabeled training examples (sentences) with the highest scores are added to the labelled training set in each iteration, and the two classifiers are trained on the new training data.

Machine learning methods have been shown to be very effective and successful in single and multi-document summarization, specifically in class specific summarization where classifiers are trained to locate particular type of information such as scientific paper summarization [51, 52, 71] and biographical summaries [62,
66, 81].

2.4 EVALUATION

The work of evaluating a summary is challenging since there is no ideal summary for a document or a group of papers, and the definition of a good summary is, to a considerable extent, an unresolved subject [58]. Human summarizers have been discovered to have a low level of consensus when it comes to analysing and writing summaries. Furthermore, the widespread use of diverse metrics, as well as the lack of an uniform evaluation criterion, has made summary evaluation complicated and challenging.

2.4.1 Evaluation of Automatically produced Summaries

Since the late 1990s, there have been various evaluation initiatives in the United States [58]. SUMMAC (1996-1998) [40], DUC (Document Understanding Conference, 2000-2007) [49], and TAC (TextAnalysis Conference, 2008-present). These conferences play a key role in the development of evaluation standards and the evaluation of summaries based on both human and machine scoring.

We must overcome three significant challenges in order to perform automatic summary evaluation: I It's critical to decide and indicate which sections of the original text should be preserved. ii) Because this information can be conveyed using a variety of phrases, evaluators must automatically recognise these bits of relevant information in the candidate summary. iii) The summary's readability in terms of grammaticality and coherence must be assessed.

2.4.2 Human Evaluation

The most straightforward way to assess the quality of a summary is to have a human do so. In DUC, for example, the judges would assess the summary's coverage, or how well the candidate summary covered the original input. Query- based summaries have been developed in more modern paradigms, namely TAC. The judges then assess how well a summary responds to the supplied question. Grammaticality, non-redundancy, integration of the most significant bits of information, organisation, and coherence are all characteristics that human specialists must examine when scoring each candidate summary. See [58] for more information.

2.4.3 Automatic Evaluation Methods.

There has been a set of metrics to automatically evaluate summaries
since the early 2000s. ROUGE is the most widely used metric
for automatic evaluation.

2.4.3.1 *ROUGE.* Recall-based measures were introduced by Lin [36].
ROUGE stands for Oriented Understudy for Gisting Evaluation.
compare the quality of a summary to that of a human
summaries (reference) ROUGE comes in a variety of forms.
(See [36]), and we'll only highlight a few of the most commonly utilised ones here:

- ROUGE-n : This metric is a recall-based statistic based on n-gram comparisons. The reference summaries and the candidate summary (automatically generated summary) evoke a succession of n-grams (usually two and three, but rarely four). Let p denote "the number of common n-grams between the candidate and the reference summary," and q denote "the number of n-grams extracted solely from the reference summary." The following formula is used to calculate the score:

$$\text{ROUGE-n} = p/q \tag{10}$$

- ROUGE-L : Between the two text sequences, this measure uses the concept of longest common subsequence (LCS). The longer the LCS between two summary phrases, the more similar they are, it is assumed. This metric is more flexible than the preceding one, but it has the disadvantage of requiring all n-grams to be consecutive. See [36] for more information on this metric and its enhanced version.

- ROUGE - SU : This metric, known as skip bi-gramand unigram ROUGE, considers both bi-grams and uni-grams. This metric allows for the insertion of words between the first and end words of the bi-grams, thus they don't have to be in order.

3. Methodology

Here we will Discuss the idea behind the implementation of text summarization that is used in this thesis. Before starting with the idea we will discuss some basic things that need to be understood so the idea of these implementations can be clearly understood.

In this thesis we have used the BERT model for text summarization. Before going ahead we will first discuss the BERT model.

BERT stands for Bidirectional Encoder Representation from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is aimed to condition both left and right context in all levels to pretrain deep bidirectional representations from unlabeled text. As a result, the pre-trained BERT model may be finetuned with just one additional output layer to provide state-of-the-art models for a variety of tasks, such as question answering and language inference, without requiring significant task-specific architecture changes.

BERT is both conceptually and empirically simple. It achieves new state-of-the-art results on eleven natural language processing tasks, including increasing the GLUE score to 80.5 percent (7.7% absolute improvement), MultiNLI accuracy to 86.7 percent (4.6%), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement), and SQuAD v2.0 Test F1 to 83.1 percent absolute improvement.

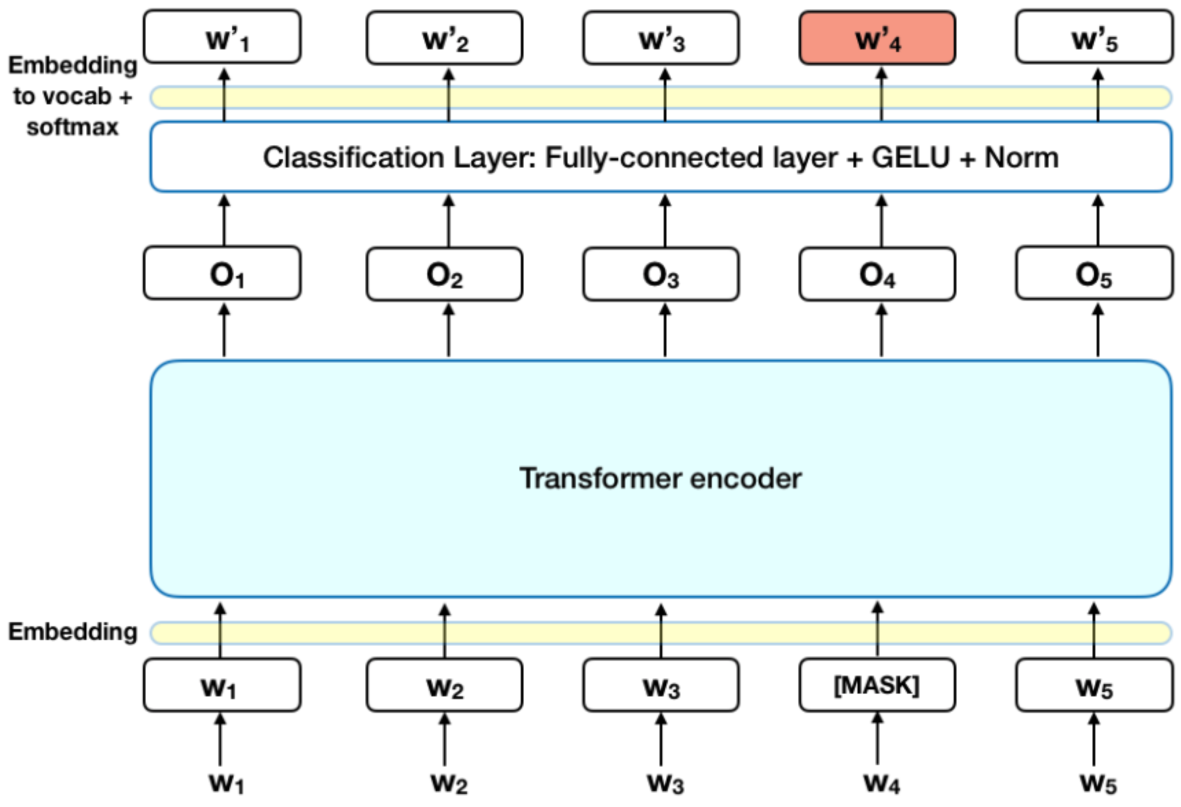
The use of the bidirectional training of Transformer, a popular attention model, to language modelling is BERT's fundamental technical breakthrough. In contrast to earlier research, which looked at a text sequence from left to right or a combination of left-to-right and right-to-left training, this study looked at a text series from left to right.The [71] results suggest that bidirectionally trained language models can have a better sense of context and flow than single-direction language models. The researchers describe a new technique called Masked LM (MLM) that allows bidirectional training in models that were previously impossible to train in.

3.1 How BERT WORKS ?

Transformer, an attention mechanism that learns contextual relationships between words (or sub-words) in a text, is used by BERT. Transformer incorporates two different mechanisms in its basic form: an encoder that reads the text input and a decoder that generates a job prediction. Only the encoder technique is required because BERT's purpose is to construct a language model. The inner workings of Transformer are outlined in a Google paper[71].

The Transformer encoder reads the complete sequence of words at once, unlike directional models that read the text input sequentially (left-to-right or right-to-left). As a result, it is classified as bidirectional, however it is more correct to describe it as non-directional. This property enables the model to deduce the context of a word from its surroundings (left and right of the word).

The Transformer encoder is described in detail in the diagram below. The input consists of a series of tokens that are embedded into vectors before being processed by the neural network. The result is an H-dimensional series of vectors, each of which corresponds to an input token with the same index. The task of defining a prediction target when training language models is difficult. Many algorithms predict the next word in a sequence (for example, "The child returned from ___"), which is a directed approach that limits context learning. BERT employs two training methodologies to meet this challenge:



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

MASKED LM (MLM)

15 percent of the words in each word sequence are substituted with a [MASK] token before being fed into BERT. Based on the context provided by the other, non-masked words in the sequence, the model then attempts to predict the original value of the masked words. In technical terms, output word prediction necessitates:

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.

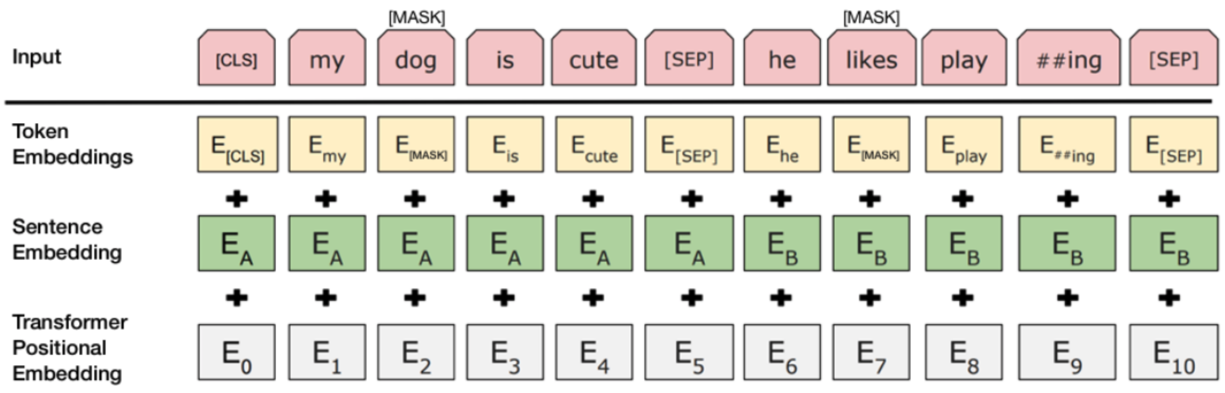
Only the prediction of masked values is taken into account by the BERT loss function, which ignores the prediction of non-masked words. As a result, the model converges more slowly than directed models, however this is counterbalanced by its improved context awareness.

3.2 NEXT SENTENCE PREDICTION (NSP)

The BERT training approach involves feeding the model pairs of sentences and learning to predict whether the second sentence in the pair is the next sentence in the original document. During training, 50 percent of the inputs are a pair in which the second sentence is the following sentence in the original text, while the other 50 percent are a random sentence from the corpus. The random sentence will, it is assumed, be detached from the first sentence.

Before entering the model, the input is processed in the following fashion to assist the model distinguish between the two sentences in training:

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 embeddings 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
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
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}

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

1. The entire input sequence go through the Transformer model.
2. The output of the [CLS] token is transformed into a 2*1 shaped vector, using a simple classification layer(learned matrices of weights and biases).
3. Calculating the probability of IsNextSequence with softmax.

Masked LM and Next Sentence Prediction are coupled for training the BERT model, with the goal of minimising the combined loss function of the two techniques.

3.3 How to use BERT(Fine Tuning)

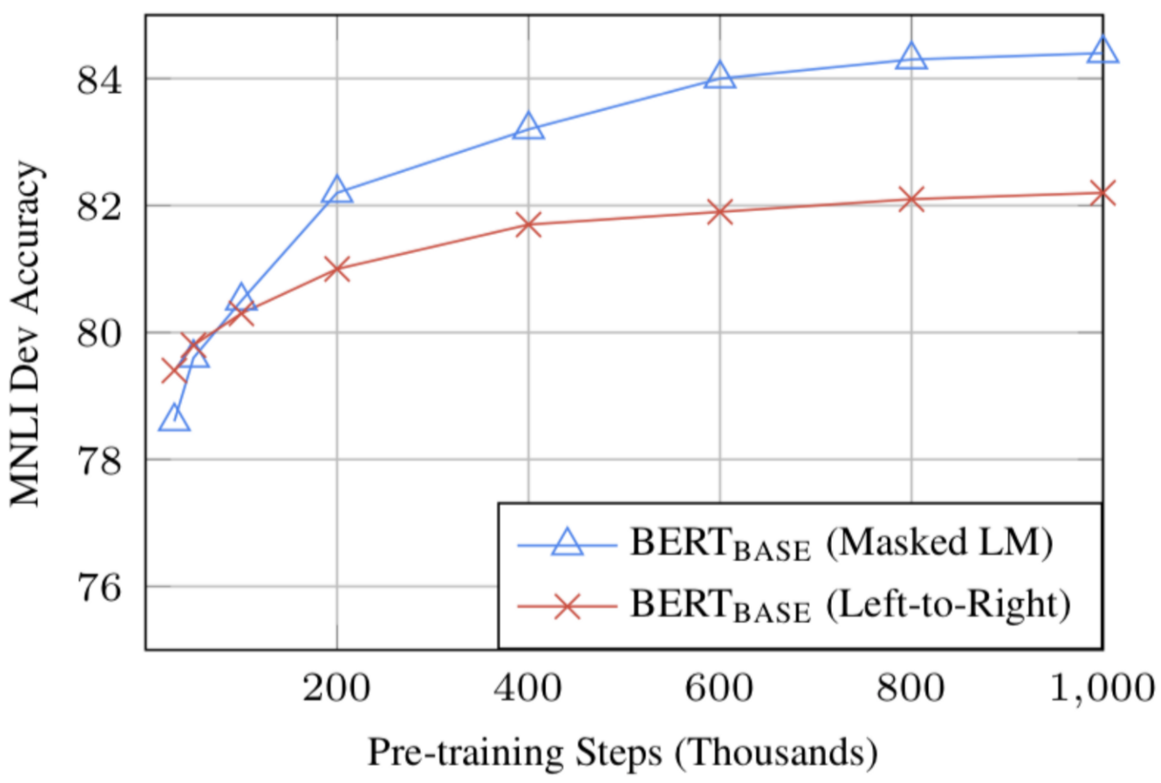
Using BERT for a specific task is relatively straight forward:

BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:

1. Sentiment analysis and other classification tasks are performed in the same way as Next Sentence classification by layering a classification layer on top of the Transformer result for the [CLS] token.
2. Question Answering tasks (such as SQuAD v1.1) need the programme to receive a question about a text sequence and mark the answer in the sequence. A Q&A model can be trained using BERT by learning two extra vectors that mark the start and finish of the answer.
3. The software in Named Entity Recognition (NER) is given a text sequence and is asked to identify the various sorts of entities (Person, Organisation, Date, etc.) that appear in it. By feeding the output vector of each token into a classification layer that predicts the NER label, a NER model may be trained using BERT.

Most hyper-parameters in fine-tuning training are the same as in BERT
training, and the paper[71] provides detailed instructions on the hyper-parameters that need to be tuned. The BERT team has used this technique to produce cutting-edge outcomes on a range of difficult natural language tasks, which are reported in Section 4 of the study.

1. Even at such a large scale, model size counts. BERT large is the largest model of its kind, with 345 million parameters. It outperforms BERT base, which utilises the similar architecture but has "only" 110 million parameters, for small-scale workloads.
2. More training steps equate to increased accuracy when there is enough training data. For instance, on the MNLI task, training on 1M steps (128,000 words batch size) results in a 1.0 percent increase in BERT base accuracy compared to 500K steps with the same batch size.
3. Because only 15% of words are predicted in each batch, BERT's bidirectional technique (MLM) converges more slowly than left-to-right approaches, but after a few pre-training steps, bidirectional training still outperforms left-to-right training.



3.4 BERT ARCHITECTURE

1. BERT base

   In the BERT base model it has 12 transformer layers along with 12 attention layers and 110 million parameters.
2. In the BERT large model it has 24 transformer layers along with 16 attention layers and 340 million parameters.

Transformer layer—A complete set of encoder and decoder layers as well as the intermediate connections make up the transformer layer. Each encoder has an RNN and Attention layers. The decoder has the same architecture as the seq2seq model, but it adds an additional attention layer between the two. Concentrating on key words is beneficial.

Summarization layers - The Self attention layer is the only significant distinction between RNN and BERT. The model aids in representation by attempting to determine the strongest connections between the words.

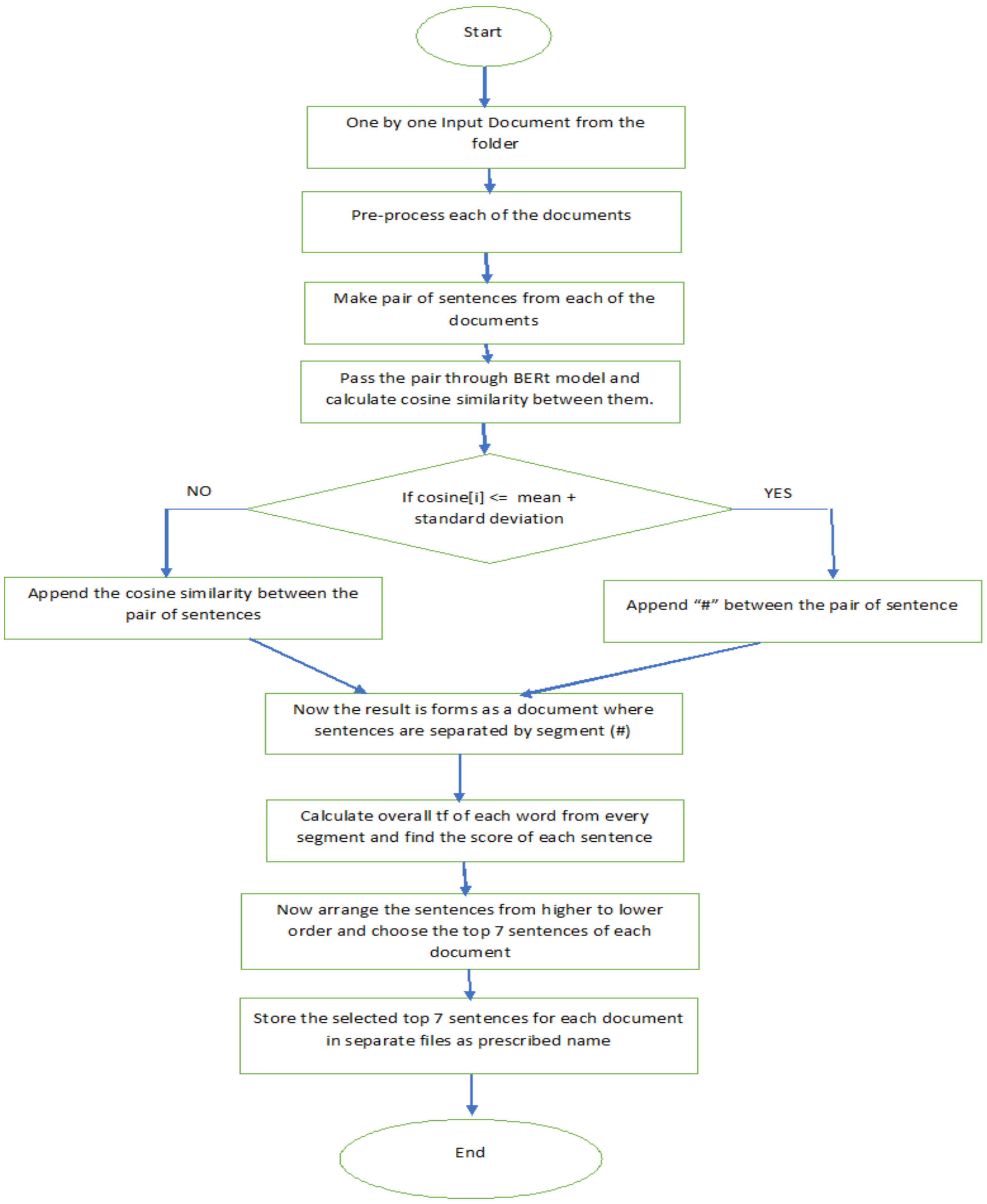
We can have different types of layers within the BERT model each having its own specifications.

1. Simple Classifier - In a simple classifier, a linear layer is added to the BERT along with a sigmoid function to predict the score Y_i.

   $$Y_i = \sigma(W_0 T_i + b_0)$$
2. Inter Sentence Transformer - In the Inter Sentence transformer, simple classifiers are not used. Rather various transformer layers are added into the model only on the sentence representation thus making it more efficient. This helps in recognizing the important points of the document.

As we have discussed the general idea behind the BERT now we will see how we have implemented the text summarization with help of bert. We have implemented it on a total number of 567 documents that are stored in the "DUC_2002_S" name folder. It has been implemented on google colab which is free to use and has a RAM of 12 per user and 108 gb disk allocated for each user from that only 77 GB of disk is usable.

Flow Chart for the implementation of Text Segmentation using BERT for Text Summarization


Start
One by one Input Document from the folder
Pre-process each of the documents
Make pair of sentences from each of the documents
Pass the pair through BERt model and calculate cosine similarity between them.
If cosine[i] <= mean + standard deviation
NO
YES
Append the cosine similarity between the pair of sentences
Append "#" between the pair of sentence
Now the result is forms as a document where sentences are separated by segment (#)
Calculate overall tf of each word from every segment and find the score of each sentence
Now arrange the sentences from higher to lower order and choose the top 7 sentences of each document
Store the selected top 7 sentences for each document in separate files as prescribed name
End

Now from the folder named “DUC_2002_S” we will select one document. After selecting the document, we will preprocess the document and remove all the unnecessary garbage that is not required for the implementation purpose. And then convert the document into a collection of sentences such as S_1, S_2, S_3,.......,S_n. There is a NLTK toolkit named “sentence tokenizer”, which helps us to create a collection of sentences from the document. Now after converting into a collection of sentences, create a pair of sentences from the collection of Sentences such as (S_1, S_2), (S_2, S_3), (S_3, S_4), ……, (S_{n-1}, S_n).

After creating the pair of sentences for each document, pass the pair through pre-trained bert mode of “bert-base-nli-mean-tokens” which is imported from the SentenceTransformer. After completing the computation process of BERT for the pair of the sentences, find the cosine similarity(Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance, chances are they may still be oriented closer together. The smaller the angle, the higher the cosine similarity.) between the pair of sentences and store it in a temporary table named pair 1, pair2, cosine similarity such as:

Pair 1	Pair 2	Cosine similarity
S_1	S_2	0.8920
S_2	S_3	0.9102
. . . .	. . . .	. . . .
S_{n-1}	S_n	0.7532

Table 1.1

After making a temporary table for each document, find the mean(Mean is the average of the given numbers and is calculated by dividing the sum of given numbers by the total number of numbers.) of the cosine similarity and standard deviation(The standard deviation is a statistic that measures the dispersion of a dataset relative to its mean and is calculated as the square root of the variance.) of the cosine similarity.

Now we have found the cosine similarity, mean of cosine similarity and standard deviation of each document. Now create an empty string and store the pair of sentences and the mean in such a way:
S1, cosine similarity of(S1,S2), S2, cosine similarity of (S2, S3), S3, cosine similarity of (S3, S4), S4,............, cosine similarity of (Sn-1, Sn), Sn. After storing this format in a string replace the cosine similarity of (S_{i-1}, S_i) with “#” where this condition satisfies:

$$cosine\ of(S_{i-1}, S_i) <= \mu - \delta$$

Where μ = mean of that cosine similarity and δ = standard deviation of the cosine similarity.

After this, the new string is stored in something like this way:

S_1 # S_2 S_3 S_4 #....... S_n.

Now here the "#" symbol denotes how many segments are there in the new string and store it in a text file. After all this we will compute the score of each sentence and rank them with higher score at first and lower score at last for each document. And from that we only select the first 7 sentences which have a high score for each document.
To calculate the score we first calculate the intra TF value and the inter TF value.
Intra TF - The number of occurence in which how many times a particular word occurs in the current sentence of that document.
Inter TF - The number of occurrences in which how many times a particular word occurs in the current document.

After calculating the inter TF and intra TF we will calculate the overall TF for every word with different values of alpha.

$$over\ all\,TF = \alpha * Inter - TF + (1 - \alpha) * Intra - TF$$

After computing the Over all TF, we will compute the weight of each word using the formula

$$weight\ of\ word = overall\,TF * IDF\ value\ of\ that\ word$$

After calculating the weight of the word, create a list of all the weights of the word, and find the mean and standard deviation for the list of weight of the word.
And calculate the sum of mean and standard deviation as value. When calculating weight of word check if weight of word is greater than equal to value then add it to the sum of overall value else weight of word equals to zero.

Now create a list that stores the sum of overall TF for each sentence and find the maximum from that list and store as maxx. And now apply the score formula for each sentence.

$$Score(S_i) = \frac{sum\ of\ overall\ sentence\ TF}{maxx} + \frac{1}{\sqrt{i}}$$

After calculating the score of each sentence, store all the sentences from higher score to lower score. And select only the top seven sentences from the document and store it in another file as the prescribed name for all the 567 documents. We have to implement all the 567 documents for alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7. That means total output file generated for this work is 567 * 7 = 3969 files.

All the output files contain Text Summary for the given document. Means for each document the output file selects only those sentences which have the main intention of the meaning of the whole document.
Thus in this way we have implemented the Text Segmentation using BERT for Text Summarization.

In the next section we will see the Implementation and the Code Part of this work.

4. Implementation and Code

```
import numpy as np
import pandas as pd
import string
import math
import pickle
import os
import json
import statistics


from google.colab import drive
drive.mount("/content/gdrive")

import os
entries = os.listdir('/content/gdrive/MyDrive/DUC_2002_S')

import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize

def word_count_inter(strr1):
    counts1 = dict()
    words = strr1.split()
    for word in words:
      if word in counts1:
        counts1[word] += 1
      else:
        counts1[word] = 1
    return counts1


def isfloat(num):
    try:
      if num is None:
        return False
      float(num)
      return True
    except ValueError:
        return False

def isNone(n):
  if n is None:
    return True
  else:
    return False

pip install sentence-transformers

cn=0
for mt in range(len(entries)):
  mm=entries[mt]
  str1 = ""
  lst2 = []
  lst1 = []
  with open ('/content/gdrive/MyDrive/DUC_2002_S/' +  mm, 'r') as fid:
    cf=fid.read().split("@@")
    ls=[]
    for line in cf:
     if(line!=""):
      xv=line
      cv=xv.split("#")
      if(len(cv)!=2):
        pass
      else:
         ls.append((cv[1]))

    for i in range(len(ls)):
      if i == len(ls) - 1:
        continue
      else:
        for j in range(i+1,i+2):
          lst2.append(ls[i])
          lst1.append(ls[j])
```

```
pairing = pd.DataFrame(
    {'pair1': lst2,
    'pair2': lst1
  })

from sentence_transformers import SentenceTransformer
model = SentenceTransformer('bert-base-nli-mean-tokens')
#Encoding
sen_embedding = model.encode(lst2)

sen_embedding1 = model.encode(lst1)
sen_embedding1.shape

from sklearn.metrics.pairwise import cosine_similarity
cosine_sim = cosine_similarity(
  sen_embedding,
  sen_embedding1
)


bert_cosine = np.empty([10000])
for i in range(len(sen_embedding)):
  bert_cosine[i] = cosine_sim[i,i]
  pairing.loc[i,'bert_cosine'] = cosine_sim[i,i]
  pairing.head()

mean = pairing["bert_cosine"].mean()
sd = pairing['bert_cosine'].std()

str2 = ""
for i in range(len(lst2)):
  n = bert_cosine[i]
  if n <= mean + sd:
    s = '#'
    str2 += lst2[i]+ " " + s + " "
  else:
    s = str(n)
    str2 += lst2[i] + "\t" + s + "\t"

from nltk.tokenize import sent_tokenize
asa = []
asa = str2.split('#')

f = open(r"/content/gdrive/MyDrive/df_final.txt", encoding = "utf-8")
asaa=[]
asaa=f.readlines()

idf_dict = {}
for line in asaa:
  key, value = line.split('#')
  idf_dict[key] = value

s = ""
for i in range(len(asa)):
  s += asa[i]

ss = s.split()
s = ""
for i in range(len(ss)):
  if isfloat(ss[i]):
    continue
  else:
    s += ss[i] + " "




s1 = sent_tokenize(s)
s2 = ""
wrd_count_intra = dict()
wrd_count_intra = word_count_inter(s)
lst = []
l = 0
```

```
sent_mapp = dict()
lst44 = []
for i in range(len(s1)):

  word_cnt_intr = dict()

  s2 = s1[i]
  if isfloat(s2):
    continue
  else:
    word_cnt_intr = word_count_inter(s2)
  ii = s2.split(" ")

  ovr_all_tf = 0.00
  sum_ovr_all_tf = 0.00
  alpha = 0.5
  wt_of_word = 0.00
  maxxx = dict()
  score_sntnc = 0.00



  for j in range(len(ii)):
    if isNone(word_cnt_intr.get(ii[j])) or isNone(wrd_count_intra.get(ii[j])):
      continue
    else:
      ovr_all_tf = alpha*word_cnt_intr.get(ii[j]) + (1-alpha)*wrd_count_intra.get(ii[j])
      idf = idf_dict.get(ii[j])
      if idf == None:
        idf = 3
      else:
        wt_of_word = ovr_all_tf*float(idf)
        lst44.append(wt_of_word)
mean2 = statistics.mean(lst44)
sd2 = statistics.stdev(lst44)
value = mean2 + sd2
for i in range(len(s1)):
  word_cnt_intr = dict()
  # Seperate Sentences from here
  s2 = s1[i]
  if isfloat(s2):
    continue
  else:
    word_cnt_intr = word_count_inter(s2)
  ii = s2.split(" ")

  ovr_all_tf = 0.00
  sum_ovr_all_tf = 0.00
  alpha = 0.5
  wt_of_word = 0.00
  maxxx = dict()
  score_sntnc = 0.00
  for j in range(len(ii)):
    if isNone(word_cnt_intr.get(ii[j])) or isNone(wrd_count_intra.get(ii[j])):
      continue
    else:
      ovr_all_tf = alpha*word_cnt_intr.get(ii[j]) + (1-alpha)*wrd_count_intra.get(ii[j])
      #print(ovr_all_tf)
      idf = idf_dict.get(ii[j])
      if idf == None:
        idf = 3
      else:
        wt_of_word = ovr_all_tf*float(idf)
        if(wt_of_word>=value):
          sum_ovr_all_tf += wt_of_word
        else:
          wt_of_word=0
          sum_ovr_all_tf += wt_of_word
  lst.append(sum_ovr_all_tf)
max_v=max(lst)

  score_sntnc = lst[i]/max_v + (1/(math.sqrt(i+1)))
  sent_mapp[s2] = score_sntnc
```

```
def sort_dict_by_value(d, reverse = False):
  return dict(sorted(d.items(), key = lambda x: x[1], reverse = reverse))

sent_mapp = sort_dict_by_value(sent_mapp,True)

s3 = ""
sub_sent_mapp = dict()
j = 0
for i in sent_mapp:
  if j == 7:
    break
  sub_sent_mapp[i] = sent_mapp.get(i)
  j = j + 1
lst33 = list(sub_sent_mapp.keys())
str33 = ""
for i in range(len(lst33)):
  str33 += lst33[i] + "\n"

stf=mm
ss=stf.partition('.txt')
bf=ss[0].partition('_')
aa=bf[0][-1].capitalize()
zx=bf[0][:-1]
vv=zx
finl=vv+"."+"P"+"."+"100"+"."+bf[2]+"."+aa+"."+"401"+".spl"
print(finl)
with open('/content/gdrive/MyDrive/output(0.6)/'+finl, 'w') as f:
  f.write(json.dumps(sub_sent_mapp))
  print("dictionary write done")
with open('/content/gdrive/MyDrive/output1(0.6)/'+finl, 'w') as fc:
  fc.write((str33))
  print("file write done")
cn+=1
```

5. Results

The result of this work is calculated by the “ROUGE” , a tool to evaluate the automatic summarization of texts as well as machine translations.

ROUGE stands for Recall - Oriented Understudy for Gisting Evaluation. Its sets metrics for evaluating automatic summarization of texts as well as machine instructions.

Here we have used ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-W, ROUGE-S4, and ROUGE -SU4 for evaluating purposes.

ROUGE-1 : It refers to the overlap of unigram (each word) between the system and reference summaries.
ROUGE - 2 :It refers to the overlap of bigrams between the system and reference summaries.
ROUGE - L : It measures the longest matching sequence of words using LCS.
ROUGE - W : Weighted LCS-based statistics that favours consecutive LCSes

Summary generated by the model has been compared with reference summary
using the ROUGE package and results are given below.

The following table shows the results:

	AVERAGE_R	**AVERAGE_P**	**AVERAGE_F**
ROUGE-1	0.47112	0.46753	0.46922
ROUGE-2	0.21105	0.20924	0.21010
ROUGE-L	0.33532	0.33274	0.33396
ROUGE - W	0.09671	0.24176	0.13813
ROUGE - S4	0.18328	0.18169	0.18244
ROUGE -SU4	0.23203	0.23010	0.23101

Table 1.2

6. CONCLUSION

Text summarization is an interesting research topic among the NLP community that helps produce concise information. The idea of this thesis is to present the latest research and progress made in this field with the Text Segmentation and BERT model method. With this method, it is proven that it can provide a more structured, broad, and diverse review, ranging from trends/topics, datasets, preprocessing, features, approach techniques, problems, methods to evaluation that are
available as a guide for future work, the relationship between trends/topics, problems and challenges in each topic, technique, and method used is summarised into one to make it easier to explore and re-analyze.

REFERENCES

[1] Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based of scientific papers. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 500–509.

[2] Rasim M Alguliev, Ramiz M Aliguliyev,Makrufa S Hajirahimova, and Chingiz A Mehdiyev. 2011. MCMR:Maximum coverage and minimum redundant text Summarization model. Expert Systems with Applications 38, 12 (2011), 14514–14522.

[3] Rasim M Alguliev, Ramiz M Aliguliyev, and Nijat R Isazade. 2013. Multiple documents summarization based on evolutionary optimization algorithms. Expert Systems with Applications 40, 5 (2013), 1675–1689.

[4] Mehdi Allahyari and Krys Kochut. 2015. Automatic topic labelling using ontology-based topic models. In Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on. IEEE, 259–264.

[5] Mehdi Allahyari and Krys Kochut. 2016. Discovering Coherent Topics with Entity TopicModels. InWeb Intelligence (WI), 2016 IEEE/WIC/ACMInternational Conference on. IEEE, 26–33.

[6] MehdiAllahyari and Krys Kochut. 2016. Semantic Context-Aware Recommendation via TopicModels Leveraging Linked Open Data. In International Conference on Web Information Systems Engineering. Springer, 263–277.

[7] Mehdi Allahyari and Krys Kochut. 2016. Semantic Tagging Using Topic Models Exploiting Wikipedia Category Network. In Semantic Computing (ICSC), 2016 IEEE Tenth International Conference on. IEEE, 63–70.

[8] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. 2017. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. ArXiv e-prints (2017). arXiv:1707.02919.

[9] Einat Amitay and Cécile Paris. 2000. Automatically summarising web sites: is there a way around it?. In Proceedings of the ninth international conference on Information and knowledge management. ACM, 173–179.

[10] Elena Baralis, Luca Cagliero, Saima Jabeen, Alessandro Fiori, and Sajid Shah. 2013. Multi-document summarization based on the Yago ontology. Expert Systems with Applications 40, 17 (2013), 6976–6984.

[11] Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to

extract and compress. In Proceedings of the 49th Annual Meeting of the Assoociation for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 481–490.

[12] DavidM Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet Allocation. the Journal of machine Learning research 3 (2003), 993–1022.

[13] Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 815–824.

[14] Yllias Chali and Shafiq R Joty. 2008. Improving the performance of the random walk model for answering complex questions. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers. Association for Computational Linguistics, 9–12.

[15] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, and others. 2006. Semi-Supervised learning. Vol. 2. MIT press Cambridge.

[16] Ping Chen and RakeshVerma. 2006. A query-based medical information Summarization system using ontology knowledge. In Computer-Based Medical systems 2006. CBMS 2006. 19th IEEE International Symposium on. IEEE, 37–42.

[17] Freddy Chong Tat Chua and Sitaram Asur. 2013. Automatic Summarization of Events from Social Media.. In ICWSM.

[18] John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden mark ov models. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 406–407.

[19] Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 305–312.

[20] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. JASIS 41, 6 (1990), 391–407.

[21] J-Y Delort, Bernadette Bouchon-Meunier, and Maria Rifqi. 2003. Enhanced web document summarization using hyperlinks. In Proceedings of the fourteenth

ACM conference on Hypertext and hypermedia. ACM, 208–215.
[22] Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. Computational linguistics 19, 1 (1993), 61–74.
[23] Harold P Edmundson. 1969. New methods in automatic extracting. Journal of the ACM (JACM) 16, 2 (1969), 264–285.
[24] Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical Centrality as salience in text summarization. J. Artif. Intell. Res.(JAIR) 22, 1 (2004), 457–479.
[25] Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In Proceedings of the 24th annual International ACM SIGIR conference on Research and development in information retrieval ACM, 19–25.
[26] Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. Journal of Emerging Technologies in Web Intelligence 2, 3 (2010), 258–268.
[27] Ben Hachey, GabrielMurray, and David Reitter. 2006. Dimensionality reduction aids term co-occurrence based multi-document summarization. In Proceedings of the workshop on task-focused summarization and question answering. Association for Computational Linguistics, 1-7.
[28] John Hannon, Kevin McCarthy, James Lynch, and Barry Smyth. 2011. personalised and automatic social summarization of events in video. In Proceedings of the 16th international conference on Intelligent user interfaces interfaces. ACM, 335–338.
[29] Sanda Harabagiu and Finley Lacatusu. 2005. Topic themes for multi-document summarization. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 202-209.
[30] Leonhard Hennig, Winfried Umbrath, and Robert Wetzker. 2008. An ontology based approach to text summarization. In Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on, Vol. 3

IEEE, 291–294.
[31] Meishan Hu, Aixin Sun, and Ee-Peng Lim. 2007. Comments-oriented blog summarization by sentence extraction. In Proceedings of the sixteenth

ACM conference on Conference on information and knowledge management. ACM, 901–904.

[32] Meishan Hu, Aixin Sun, and Ee-Peng Lim. 2008. Comments-oriented document summarization: understanding documents with readers' feedback. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 291–298.

[33] Kevin Knight and DanielMarcu. 2000. Statistics-based summarization-step one: Sentence compression. In AAAI/IAAI. 703–710.

[34] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The Annals of Mathematical Statistics (1951), 79–86.

[35] Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In Proceedings of the 18th annual international ACM SIGIR on conference Research and development in information retrieval. ACM, 68–73.

[36] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text Summarization Branches Out: Proceedings of the ACL-04 Workshop. 74–81.

[37] Elena Lloret and Manuel Palomar. 2012. Text summarisation in progress: a literature review. Artificial Intelligence Review 37, 1 (2012), 1–41.

[38] Hans Peter Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of research and development 2, 2 (1958), 159–165.

[39] Inderjeet Mani and Eric Bloedorn. 1999. Summarising similarities and difference among related documents. Information Retrieval 1, 1-2 (1999), 35–67.

[40] Inderjeet Mani, Gary Klein, David House, Lynette Hirschman, Therese Firmin, and Beth Sundheim. 2002. SUMMAC: a text summarization evaluation. Natural Language Engineering 8, 01 (2002), 43–68.

[41] QiaozhuMei and ChengXiang Zhai. 2008. Generating Impact-Based Summaries for Scientific Literature.. In ACL, Vol. 8. Citeseer, 816–824.

[42] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. Association for Computational Linguistics.

[43] Rada Mihalcea and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization. (2005).

[44] Liu Na, Li Ming-xia, Lu Ying, Tang Xiao-jun, Wang Hai-wen, and Xiao Peng. 2014. Mixture of topic model for multi-document summarization. In Control and Decision Conference (2014 CCDC), The 26th Chinese. IEEE, 5168–5172.

[45] Ani Nenkova and Amit Bagga. 2004. Facilitating email thread access by extractive summary generation. Recent advances in natural language processing III: selected papers from RANLP 2003 (2004), 287.

[46] Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In Mining Text Data. Springer, 43–76.

[47] Paula S Newman and John C Blitzer. 2003. Summarising archived discussions: a beginning. In Proceedings of the 8th international conference on Intelligent user interfaces. ACM, 273–276.

[48] You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models
to query-focused multi-document summarization. Information Processing & Management 47, 2 (2011), 227–237.

[49] Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. Inf. Process. Manage. 43, 6 (Nov. 2007), 1506–1520. https://doi.org/10.1016/j.ipm.2007.01

[50] Makbule Gulcin Ozsoy, Ilyas Cicekli, and Ferda Nur Alpaslan. 2010. Text summarization of turkish texts using latent semantic analysis. In Proceedings of the 23rd international conference on computational linguistics. Association for Computational Linguistics, 869–876.

[51] Vahed Qazvinian and Dragomir R Radev. 2008. Scientific paper summarization using citation summary networks. In Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for ComputationalLinguistics, 689–696.

[52] Vahed Qazvinian, Dragomir R Radev, Saif M Mohammad, Bonnie Dorr, David Zajic, Michael Whidby, and Taesun Moon. 2014. Generating extractive summaries of scientific paradigms. arXiv preprint arXiv:1402.0556 (2014).

[53] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Computational linguistics 28, 4 (2002), 399–408.

[54] Dragomir R Radev,Hongyan Jing, andMalgorzata Budzikowska. 2000. Centroid Based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization. Association for Computational Linguistics, 21–30.

[55] Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004.

Centroid-based summarization of multiple documents. Information Processing & Management 40, 6 (2004), 919–938.

[56] Owen Rambow, Lokesh Shrestha, John Chen, and Chirsty Lauridsen. 2004. Summarizing email threads. In Proceedings of HLT-NAACL 2004: Short Papers. Association for Computational Linguistics, 105–108.

[57] Zhaochun Ren, Shangsong Liang, Edgar Meij, and Maarten de Rijke. 2013. Personalized time-aware tweets summarization. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 513–522.

[58] Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In Multi-source, Multilingual Information Extraction and Summarization. Springer, 3–21.

[59] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. Information processing & management 24, 5 (1988), 513–523.

[60] Yogesh Sankarasubramaniam, Krishnan Ramanathan, and Subhankar Ghosh. 2014. Text summarization using Wikipedia. Information Processing & Management 50, 3 (2014), 443–461.

[61] Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. Journal of the AmericanMedical Informatics Association 17, 5 (2010), 507–513.

[62] Barry Schiffman, Inderjeet Mani, and Kristian J Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 458–465.

[63] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. 2010. Summarising microblog automatically. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 685–688.

[64] Beaux P Sharifi, David I Inouye, and Jugal K Kalita. 2013. Summarization of Twitter Microblogs. Comput. J. (2013), bxt109.

[65] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007.

Document Summarization Using Conditional Random Fields.. In IJCAI, Vol. 7. 2862–2867

[66] Sérgio Soares, Bruno Martins, and Pavel Calado. 2011. Extracting biographical sentences from textual documents. In Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011), Lisbon, Portugal. 718–30.

[67] Karen Spärck Jones. 2007. Automatic summarising: The state of the art. Information Processing & Management 43, 6 (2007), 1449–1481.

[68] Josef Steinberger, Massimo Poesio, Mijail A Kabadjov, and Karel Ježek. 2007. Two uses of anaphora resolution in summarization. Information Processing & Management 43, 6 (2007), 1663–1680.

[69] Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. Handbook of latent semantic analysis 427, 7 (2007), 424–440.

[70] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In Proceedings of the 16th international conference on the World Wide Web. ACM, 697–706.

[71] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.