TEXT SUMMARIZATION SYSTEM USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

MANEESH NARAYAN

2K21/CSE/14

Under the supervision of

DR. MANOJ SETHI

(Professor)

M.Tech (Computer Science & Engineering) **Maneesh Narayan**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

May, 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Maneesh Narayan, Roll No. 2K21/CSE/14 student of M. Tech (Computer Science and Engineering), hereby declare that the Project Dissertation titled "**TEXT SUMMARIZATION SYSTEM USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING**." which is being submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi, in partial fulfilment of requirements for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **MANEESH NARAYAN**

Date: (2K21/CSE/14)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I, hereby certify that the Project titled "**TEXT SUMMARIZATION SYSTEM USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING.",** which is submitted by Maneesh Narayan, Roll No. 2K21/CSE/14, Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

DR.MANOJ SETHI

(Professor)

SUPERVISOR

ABSTRACT

Information overload, caused by the rapid growth of the Internet, has become a significant issue. The abundance of information available online necessitates the simplification of relevant content through summarization. Manual summarization of large amounts of text is a daunting task for humans, leading to a demand for more advanced and powerful summarization techniques. Researchers have been striving to enhance the quality of machine-generated summaries to match those created by humans since the 1950s. This study presents a comprehensive analysis of text summarization, covering various aspects such as summarization approaches, techniques, standard datasets, evaluation metrics, and future research prospects.The study delves into two widely accepted approaches in text summarization: extractive and abstractive. These approaches are explored in detail, highlighting their strengths and limitations. To facilitate comparison and replication of findings, the evaluation of summaries and the development of reusable resources and infrastructure are crucial, fostering healthy competition and driving improvements in summarization outcomes. The study also discusses different evaluation methods employed to assess the quality of generated summaries.In conclusion, this study provides valuable insights into the field of text summarization. It examines the current state of the art, including summarization techniques, datasets, evaluation metrics, and future research opportunities. By summarizing vast amounts of information effectively, text summarization can alleviate the burden of information overload. The challenges and opportunities discussed in this study offer a roadmap for researchers interested in advancing text summarization techniques and contributing to this evolving field.

ACKNOWLEDGEMENT

I am extremely grateful to my project guide, Dr. Vinod Kumar, Professor, Department of Computer Science and Engineering, Delhi Technological University, Delhi for providing invaluable guidance and being a constant source of inspiration throughout my research. I will always be indebted to him for the extensive support and encouragement he provided. I am highly indebted to the panel faculties during all the progress evaluations for their guidance, constant supervision and for motivating me to complete my project work. They helped me throughout by giving new ideas, providing necessary information and pushing me forward to complete the project work.

MANEESH NARAYAN

(2K21/CSE/14)

CONTENTS

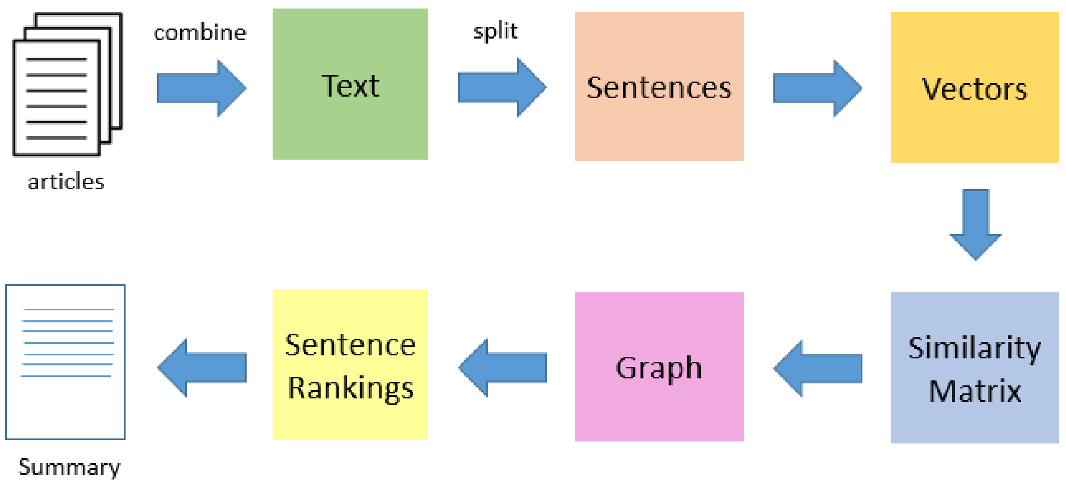
LIST OF FIGURES

CHAPTER 1 INTRODUCTION

In the age of the internet, massive volumes of textual data have been amassed and are still growing exponentially every day in the form of cloud resources including websites, blogs, news, user messages, and social networking sites. Several more articles, books, novels, court records, scientific papers, biomedical materials, and other archives also include substantial textual content. Information overload is hence becoming a greater issue.Users regularly spend a significant amount of time viewing various long texts and filtering out unnecessary information, which significantly lowers their productivity. It is now an essential and fundamental issue that needs to be solved to efficiently discover the information required from the text resources, then summarise and compress it.Manual summarization requires initial analyzing every item of content, which is very expensive and vulnerable to error in huge amounts of data. Automated text summarization (ATS) offers a practical solution to this issue.The goal of ATS is to automatically produce a brief and understandable summary that conveys the key points of the input text. The problem of how to collect necessary information promptly, accurately, and efficiently is becoming more and more important. ATS has evolved into one of the most challenging tasks in the field of natural language processing(NLP) as a result of the complexity of the input text

In early 1958, H. P. Luhn [1] publish a paper on the study of ATS.It proposed a Statistical information derived from word frequency and distribution form the magazine articles and technical papers.In M. T. Maybury[2] built a system that can choose important data from an event database, and a high-quality summary was defined as the most important information obtained from the input document.int 2002, D. R. Radev, E. Hovy, and K. McKeown [3] defined the summary as a grouping of sentences produced from a number of (or a single) input documents that includes the main topics of the input materials. They made a point of emphasising that the generated summary is only half as long as the input, if not even less. The summaries should cover the main points of the input document and be brief, which is one of the ATS tasks' key qualities that is captured in the previous descriptions.

Based on how the summaries are produced, there are typically two well-known summarization systems: extractive summarization and abstractive summarization (ABS). Extractive systems, such as graph-based approaches (such as LexRank [4]), centrality-based methods (such as Centroid [5]), and corpus-based methods, directly extract sentences or words from the original content to provide a summary . Before employing the algorithm of natural language generation (NLG) to create a more succinct summary using paraphrase, synonymous replacement, sentence compression, etc., abstractive systems must first comprehend the semantics of the text. ABS is therefore more similar to the process of handwritten summaries when compared to extractive summarization.



CHAPTER 2 LITERATURE SURVEY

Text summarization using machine learning and natural language processing (NLP) is an area of research and application that aims to automatically generate concise and coherent summaries from textual documents. It involves leveraging various machine learning techniques and NLP models to understand the content of the text, extract key information, and produce informative summaries. Let's dive deeper into the process of text summarization using machine learning and NLP.Text summarization can be broadly classified into two categories: extractive and abstractive summarization.

1. **Extractive Summarization**:
   In extractive summarization, the system selects important sentences or phrases from the source document and combines them to form a summary. It relies on identifying salient information rather than generating new text **[6]**.
2. **Abstractive Summarization**:
   Abstractive summarization aims to generate summaries by understanding the context and meaning of the source document and creating concise and coherent summaries that may include novel sentences or paraphrases **[7]**.

To train a machine learning model for text summarization, a dataset of source documents and their corresponding summaries is required. This data can be collected from various sources such as news articles, scientific papers, or online resources.

Once the dataset is collected, preprocessing is performed to clean and normalize the text. This involves removing special characters, converting text to lowercase, tokenizing the text into words or subwords, and applying techniques like stemming or lemmatization to normalize the words.

After all the preprocessing process it need represent the textual data in a suitable format for machine learning models, feature extraction techniques are applied. Common approaches include:

1. **Bag-of-Words (BoW)**: The Bag-of-Words (BoW) model is a popular technique in natural language processing (NLP) and information retrieval. It represents a document as an unordered collection of words and treats it as a "bag" of its constituent words. The model disregards grammar and word order, focusing solely on the presence or frequency of words in the document **[8]**.To implement the BoW model, several steps are involved. First, the text is tokenized by breaking it into individual words or tokens. This creates the basis for the subsequent analysis. Next, a vocabulary is created by compiling a list of unique

words present in the corpus. Each word in the vocabulary becomes a feature in the model.After creating the vocabulary, feature extraction takes place. This involves constructing a feature vector for each document in the corpus based on the vocabulary. The feature vector represents the frequency or presence of each word in the document. There are two common approaches for feature extraction: frequency-based and presence-based. In the frequency-based approach, the feature vector contains the count of each word in the document. In the presence-based approach, the feature vector consists of binary values indicating whether a word is present or absent in the document **[9]**.The final representation of a document is a high-dimensional vector, where each entry corresponds to a word in the vocabulary. The order of the words is ignored, and the vector is sparse since most entries are zero due to the sparsity of words in each document.The BoW model has several advantages. It is simple to implement and computationally efficient, making it suitable for large datasets. It is also language-independent and can be applied to a wide range of NLP tasks, such as text classification and sentiment analysis. The model effectively captures keyword information and document-level statistics.However, the BoW model has limitations. It overlooks the semantics and relationships between words in a document. Since it treats each word independently, it fails to consider the context and order of words. As a result, it may lose valuable information, especially for tasks requiring a deeper understanding of the text. To address these limitations, more advanced techniques like word embeddings and deep learning models have been developed.In summary, the Bag-of-Words model is a foundational technique in NLP that represents documents as collections of words, disregarding grammar and word order. It constructs feature vectors based on word frequencies or presence and has been widely used in various NLP tasks. While simple and effective, the model has limitations in capturing semantic information and word relationships. Researchers continue to explore more sophisticated models to enhance the representation and understanding of textual data **[10]**.

2. **TF-IDF (Term Frequency-Inverse Document Frequency)(formulas)**: Term Frequency-Inverse Document (TF-IDF) Frequency) is a well-known numerical statistic used in text mining and information retrieval to evaluate the significance of a phrase in a document or database,an corpus. Inverse document frequency and word frequency are combined in this sentence. A phrase's frequency inside a

text is measured by term frequency (TF). It serves to highlight a word's significance inside a particular documentA term's term frequency is determined by how frequently it appears in a document.The inverse document frequency (IDF) method determines a term's rarity within a corpus. By taking into account how frequently a phrase appears in the full collection of papers, it measures how informative it is. Higher IDF ratings for uncommon phrases that are found in fewer papers suggest their potential importance.The term frequency and inverse document frequency are multiplied to get the TF-IDF. **[11]**. The formula is as follows:

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ if\ terms\ in\ d}$$
$$IDF(t) = log\frac{N}{1+df}$$
$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

The resulting TF-IDF score reflects the relative importance of a term in a document or a collection of documents. If a term has a high frequency within a document but appears in many other documents as well, its TF-IDF score will be moderate. Conversely, if a term has a high frequency in a document but is rare across the corpus, its TF-IDF score will be high.TF-IDF is widely used in various natural language processing tasks, including information retrieval, document classification, and text summarization. It helps to identify important terms and distinguish them from common ones. By considering both local (term frequency) and global (inverse document frequency) aspects, TF-IDF provides a more nuanced representation of the significance of terms in a document collection.In practice, TF-IDF can be computed using various algorithms and libraries. It is often employed as a feature in machine learning models or as a ranking factor in search engines. Researchers and practitioners continue to explore variations and enhancements to TF-IDF to improve its effectiveness in different applications.

3. **Word Embeddings**: Word embeddings are a fundamental concept in natural language processing (NLP) and machine learning that enable computers to represent and understand the meaning of words **[11]**. In essence, word embeddings are dense vector representations that capture

the semantic relationships and contextual information of words in a mathematical space.Traditionally, words in NLP were represented as one-hot vectors, where each word in a vocabulary was assigned a unique index, and the vector contained all zeros except for a single one at the index corresponding to the word. However, one-hot vectors are sparse and fail to capture the inherent semantic relationships between words.Word embeddings, on the other hand, encode words as continuous-valued vectors of fixed dimensions, typically ranging from 50 to 300 dimensions. These vectors are learned by training models on large corpora of text data using unsupervised learning techniques, such as Word2Vec, GloVe, or FastText **[12]**.

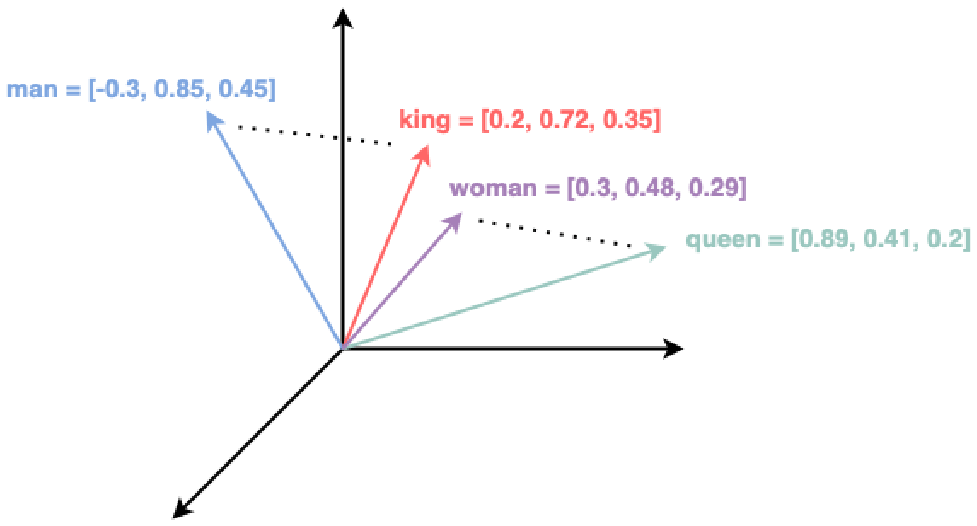


Fig 2.0 Word Embedding

The main idea behind word embeddings is the distributional hypothesis, which states that words that occur in similar contexts tend to have similar meanings. By analyzing the co-occurrence patterns of words in a text corpus, word embedding models learn to represent words that have similar meanings as vectors that are close together in the embedding space.Word embeddings capture not only the semantic relationships between words but also capture syntactic and contextual information. This allows the models to understand the meaning of words in different contexts and perform better on various NLP tasks, such as sentiment analysis, machine translation, named entity recognition, and text summarization.One of the significant advantages of word embeddings is their ability to handle out-of-vocabulary (OOV) words. Since word embeddings are learned based on the context in which words appear, they can provide meaningful representations for words that were not present in the training data. This is particularly useful in real-world applications where new words or terms continually emerge.

Word Embeddings: Word embeddings capture the semantic relationships between words by mapping them into a dense vector space. Pretrained models like Word2Vec, GloVe, or FastText can be used to obtain word embeddings **[11] [12]**.

- **Support Vector Machines (SVM)**:Support Vector Machines (SVM), a supervised machine learning technique, is frequently used for classification and regression applications. It has an excellent reputation for being adaptable and successful in managing both linear and non-linear patterns in data. Finding the best hyperplane to partition data points into discrete groups is the goal of SVM. This hyperplane serves as a decision boundary in binary classification by maximising the distance between the closest data points in each class. Support vectors, which are very important pieces of information, are necessary in establishing the hyperplane. SVM accomplishes its goal by projecting the input data onto a higher-dimensional feature space, which makes it simpler to separate the data points. This mapping is accomplished using a kernel function that measures the similarity between pairs of data points. Popular kernel functions include linear, polynomial, Gaussian radial basis function (RBF), and sigmoid. The choice of kernel function depends on the data characteristics and the specific problem at hand.Once the data is mapped into the higher-dimensional feature space, SVM aims to identify the hyperplane that maximizes the margin between the support vectors. The margin represents the distance between the hyperplane and the nearest support vectors. A larger margin indicates better generalization performance of the model. SVM is capable of handling both linearly separable and non-linearly separable data. In cases where the data is linearly separable, SVM identifies a linear hyperplane to separate the different classes. However, for non-linearly separable data, SVM employs the kernel trick to implicitly transform the data into a higher-dimensional space where it can be effectively separated using a linear hyperplane.

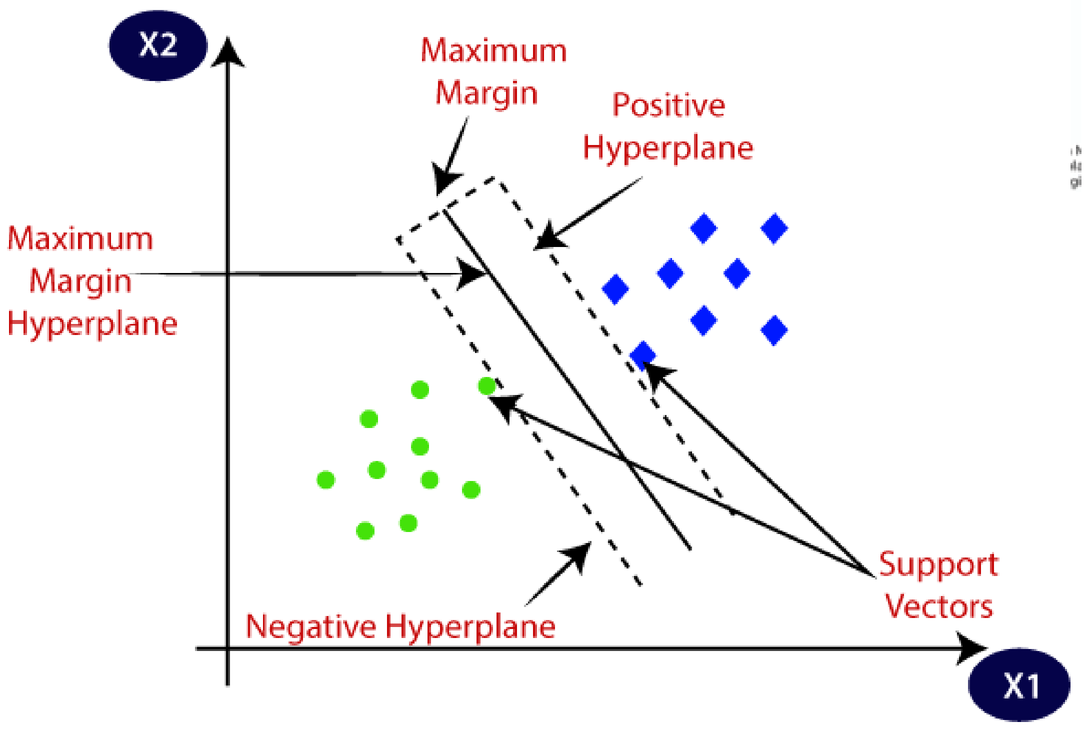


Fig 2.1 Support vector Machine

This allows SVM to handle complex decision boundaries and capture non-linear relationships in the data.In addition to classification, SVM can also be used for regression tasks. In regression, SVM finds a hyperplane that best fits the data while minimizing the error between the predicted and actual values. The objective is to find a hyperplane that lies within a certain margin around the data points.SVM has several advantages. It is effective in handling high-dimensional data, works well with small to moderate-sized datasets, and has a strong theoretical foundation. It is also less prone to overfitting compared to other algorithms. SVM can handle both numerical and categorical data through appropriate kernel functions.However, SVM has some limitations. It can be computationally expensive, especially with large datasets. The choice of the kernel function and its parameters can significantly impact the performance of the model. SVM is also sensitive to outliers as they can affect the position of the hyperplane **[14]**.

- **Graph-based Models**: Graph-based models are a category of algorithms used in text summarization that leverage the structure of the document to identify important sentences and generate summaries. These models represent the document as a graph, where sentences are nodes, and the relationships between them are represented as edges.In graph-based models, the first step is to construct the graph representation of the document. Each sentence is treated as a node, and the edges between nodes capture the relationships between sentences. These relationships can be based on various criteria, such as sentence similarity, semantic coherence, or temporal ordering.Once the graph is constructed, the importance of each sentence is determined by analyzing its centrality within the graph. Centrality measures, such as PageRank or HITS (Hyperlink-Induced Topic Search), are commonly used to assign importance scores to sentences. These measures take into account the connectivity of the nodes in the graph and the importance of neighboring sentences.After determining the importance scores, the graph-based model selects the most important sentences to form the summary. This selection process can be performed using different strategies, such as selecting sentences with the highest importance scores until a certain length constraint is reached or using optimization algorithms to find the optimal subset of sentences that maximize a predefined objective function.Graph-based models have several advantages in text summarization. They can capture the relationships and dependencies between sentences, enabling a more coherent and contextually relevant summary **[4]**.

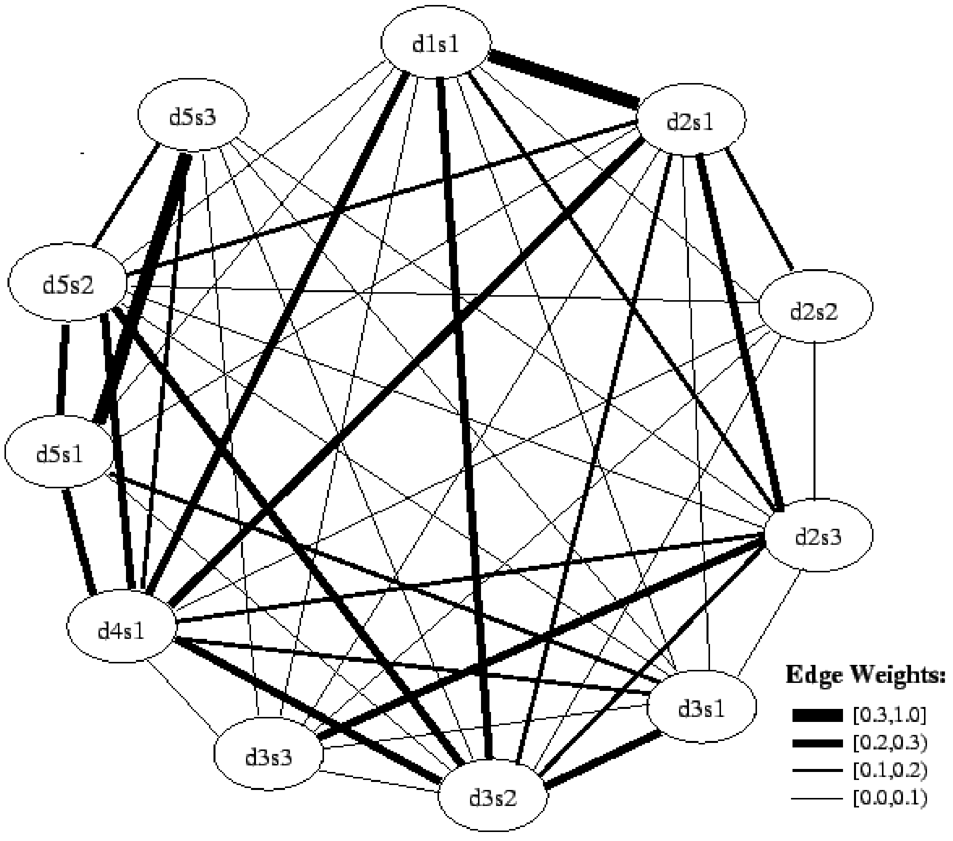


Fig 2.2 Graph-based Model

By considering the global structure of the document, they can overcome the limitations of extractive methods that may miss important information or fail to provide a cohesive summary.Additionally, graph-based models can handle different types of documents, such as news articles, scientific papers, or social media posts, by adapting the graph construction and importance scoring strategies to the specific characteristics of each domain.However, graph-based models also face challenges. Constructing an accurate and representative graph requires addressing issues like sentence representation, semantic understanding, and handling various types of relationships. Furthermore, as the size of the document increases, the graph's complexity grows, making the summarization process computationally expensive.In recent years, advancements in graph neural networks (GNNs) have further improved the capabilities of graph-based models in text summarization. GNNs can capture more complex relationships between nodes and incorporate contextual information, leading to more accurate importance scoring and better summary generation.

- **Neural Networks**: Neural networks play a significant role in text summarization, enabling the development of more advanced and accurate models. These models can be trained to understand the context of a document and generate concise and coherent summaries. Neural networks excel in learning complex patterns and representations from data, making them suitable for text summarization tasks.In the context of text summarization, neural networks can be used for both extractive and abstractive approaches. In extractive summarization, neural networks can be employed to classify sentences based on their importance or relevance to the overall content. These models learn to assign scores or probabilities to individual sentences, allowing the selection of the most salient ones for inclusion in the summary. Various architectures, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer models like BERT, have been used for this purpose.For abstractive summarization, neural networks are used to generate summaries that capture the essence of the source text. These models learn to comprehend the input document and produce new sentences that effectively summarize the information. Recurrent neural networks, such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), are commonly employed due to their ability to handle sequential data and capture contextual dependencies. Transformers, like the popular BERT (Bidirectional Encoder

Representations from Transformers), have also shown promising results in abstractive summarization by capturing long-range dependencies and improving the fluency of generated summaries.Training neural networks for text summarization typically involves large-scale datasets, including pairs of source documents and their corresponding summaries. These datasets are used to optimize the network parameters through techniques like backpropagation and gradient descent. Loss functions, such as cross-entropy or maximum likelihood estimation, are employed to measure the discrepancy between the generated summaries and the ground truth summaries. Reinforcement learning can be applied to fine-tune the models, using reward signals to encourage the generation of high-quality summaries.Neural networks for text summarization face certain challenges. Coherence and fluency are important aspects, and ensuring that generated summaries maintain the original meaning and flow can be a complex task. Additionally, training data biases and the lack of diverse datasets can impact the performance of these models. Domain adaptation is another challenge, as models trained on one domain may not generalize well to other domains with different terminologies and writing styles.In conclusion, neural networks have revolutionized text summarization by enabling both extractive and abstractive approaches. These models leverage their ability to learn patterns and representations from data to effectively summarize textual information. They have been trained using large-scale datasets and optimized through backpropagation and gradient descent. While neural networks have shown remarkable performance in text summarization, challenges related to coherence, fluency, biases, and domain adaptation persist and require ongoing research and development efforts to overcome.

Abstractive summarization involves generating summaries by understanding the content and context of the source document. Neural networks, particularly sequence-to-sequence models, are commonly used for abstractive summarization:

- Encoder-Decoder Architecture: An encoder-decoder architecture, frequently based on transformers or recurrent neural networks (RNNs). The encoder converts the source document into a fixed-length representation after processing it. The summary is subsequently produced by the decoder using the encoded representation.
- Attention Mechanism:To enhance communication between the encoder and the decoder, attention methods are used. They enable the model to

concentrate on various areas of the source document when producing the summary, assisting in the acquisition of essential data and enhancing coherence.

- Training with Ground Truth Summaries: During the training phase, the abstractive summarization model is provided with pairs of source documents and their corresponding human-generated summaries. The model's parameters are optimized to minimize the discrepancy between the generated summaries and the ground truth summaries. Techniques like cross-entropy loss or maximum likelihood estimation are commonly used as training objectives.
- Fine-tuning with Reinforcement Learning: To further improve the generated summaries, reinforcement learning techniques can be applied. Reinforcement learning uses reward signals to guide the model towards generating high-quality summaries. It encourages the model to explore different wordings and sentence structures, leading to more creative and diverse summaries.

Evaluating the quality of generated summaries is a crucial step. Various evaluation metrics can be used to measure the effectiveness of summarization systems. Common metrics include:

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE measures the overlap between the generated summary and the reference summary in terms of n-gram matches, longest common subsequences, and skip-bigram matches **[15]**.
- BLEU (Bilingual Evaluation Understudy): BLEU, which was first created for machine translation assessment, is now utilised for summarization. It calculates the n-gram overlap between the reference summary and the summary that was created. **[16]**.
- METEOR (Metric for Evaluation of Translation with Explicit Ordering): METEOR combines precision, recall, and alignment-based measures to assess the quality of generated summaries **[17]**.
- Human Evaluation:In addition to automatic metrics, human review is useful for determining the summaries' general quality, coherence, and readability. The summaries may be judged by human assessors using standards including relevance, fluency, and subject coverage.

2.1 Extractive Text Summarization

In this portion, we are limited to extractive summarization. We briefly outline the fundamental strategies and go into deeper depth on cutting-edge machine learning and graph-based strategies.

A text must be pre-processed before it can be summarized. Pre-processing reduces dimensions by reducing noise such as meaningless words or conjugation. This decrease makes managing the vectorization of documents much simpler. The pre-processing stage frequently entails the following steps: segmenting the text into chunks of phrases, sentences, and paragraphs, segmenting the chunks into words (tokenization); normalizing the words (lemmatization, stemming); removing stopwords; POS tagging identifying named entities extracting terms and keywords; and weighting terms which depend on representation. The complexity of each of these procedures varies depending on the language. A language identification module like TreeTagger can be used to determine the language if it is unidentified **[18]**.

Almost all summarizers carry out three tasks: picking a summary, scoring sentences, and producing an intermediate representation of the input.

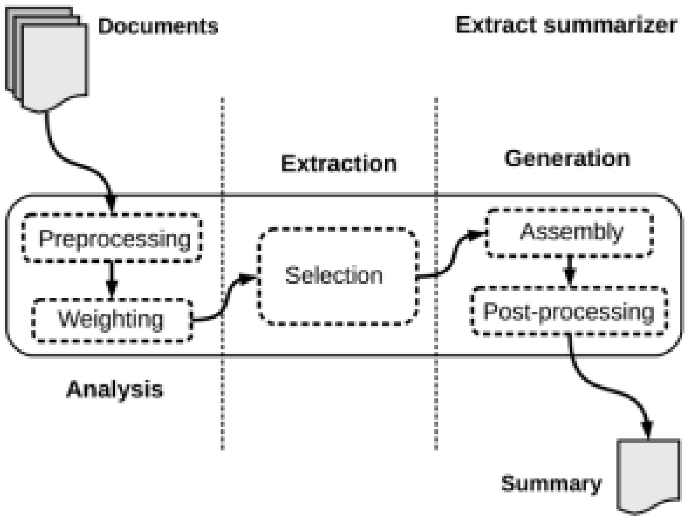


Fig 2.3 Extractive Text Summarization

For Vector Model, *Let* $t_1, t_2, t_3, ..., t_k)$ *be the set of terms and* $D_1, D_2, D_3, ..., D_N$ *the set of documents. A document* D_i *is represented as a vector:*

$$d_i = w(D_i, t_1), w(D_i, t_2), ..., w(D_i, t_k)$$

Where $w(D_i, D_j)$ *is the weight of a term* t_j *in the document* D_i

By ignoring word order, we arrive at a bag-of-words model where each word is given a weight based on its significance within the text.There are three types of weighting $\omega_{i,j}$ equals 1 if term is present in the sentence and 0 if not, occurrences or frequency (number of time a term is present in the sentences) and corrective (frequency normalized by word distribution - $TF \times IDF$).The result is a $S_{|\rho \times N|}$matrix of ρ rows (sentences) and N columns (terms in the lexicons). Each sentence s'_{μ} is projected as a vector of terms weighted by their prominence in it.

$$S = \begin{pmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,N} \\ S_{2,1} & S_{2,2} & \dots & S_{2,N} \\ \vdots & \vdots & \dots & \vdots \\ S_{\rho,1} & S_{\rho,2} & \dots & S_{\rho,N} \end{pmatrix}$$

$$S_{\mu,j} = \begin{cases} \omega_{\mu,j} & when\ the\ word\ j \in S'_{\mu} \\ 0 & otherwise \end{cases}$$

Where each μ row contains the $\omega_{\mu,j}$ weighting of word j in a S'_{μ} sentence **[19]**.

Indicator representation, Indicator representation makes use of indicators of importance for example proper nouns, sentence length, the presence of numerical data in a sentence, the location of a sentence in the document, and the presence of words in the title. Cue phrases are a list of specific phrases that identify a potentially significant sentence. Studies show that the phrase's placement in the text is the most important indicator, and that cue words, title words, and positional keywords are more effective than simply distributing the frequency of keywords **[20]**.

In a **graph representation**, the edges between the vertices (or nodes) indicate the relationships between the text units (words, phrases, or documents). This relationship can be expressed using word overlap or a cosine similarity. We can represent the relation in binary form rather than giving the edges weights; vertices are connected if their similarity is greater than a predetermined threshold.

Graph G might be undirected, directed either forward or backward, or both. A sentence in a graph that is pointed forward or backward simply indicates the sentences that come after or before it in the text. Then, a summary is created by extracting the best sentences. While backward graphs are better suited to journalistic publications, forward graphs are appropriate for cinema and literary critique. A sentence can suggest any of the two sentences in undirected graphs **[4]**.

Additionally, the summarizer chooses summary phrases. The summary's length is typically restricted. The best n strategy is used to choose the top n phrases with the highest scores. To avoid repetition, we can penalise statements that are exact duplicates of the ones that were previously chosen for a summary. Maximum Marginal Relevance (MMR) formula below maximises relevant data while minimising unnecessary data. When a new sentence is chosen, the scores are updated according to a linear combination of its starting weight and how similar it is to previously selected words.

$$\omega_{MMR(s)} = argmax_{s \in D/Sum}[\lambda \underbrace{sim_1(s, Q)}_{Relvance} - (1-\lambda) \underbrace{argmax_{S_{\mu \in Sum}} sim_2(s, s_\mu)}_{Redundancy}]$$

Where D is the set of sentence in documentm, Sum is the set of sentences already selected for the summary, Q is a query (topic of the summary), λ is a penalization coefficient and $sim_i()$ is a function which returns the similarity of two sentences.

In $globlalselection$ approaches,limitations are imposed on the subset of words to increase overall significance, reduce repetition, and increase coherence.

2.1.1 Naive Bayes

This approach of summarising was handled as a statistical classification problem. Researchers developed a Bayes classifier to calculate the likelihood that a particular text belongs in the summary. They utilized hand-selected summaries for their training set.

It used six discrete characteristics to train the classifier, which was prioritized in the following order:

1. Paragraph feature: This feature shows where a sentence falls in a paragraph. The authors noted that a sentence's significance for a summary might vary depending on where it is in a paragraph.
2. Fixed-phrase feature: This feature determines if a certain phrase from a predetermined list is included in the sentence. These words and phrases might help determine how important a sentence is for summary extraction.
3. Sentence length cutoff feature: This function establishes a limit, designated as $u1$, with a value of 5. Over this length threshold, sentences are deemed significant for a summary.

4. Thematic word feature: The existence of thematic words in the sentence is determined by this attribute. Words that are directly connected to the text's principal theme or topic are known as thematic words. Such keywords can be used to determine if a statement is appropriate for summary creation **0**.
5. Uppercase word feature: This characteristic indicates if the phrase contains words in capital letters. It noted that capitalized words frequently denote significant individuals or essential details in the text, which may add to the relevance of the phrase for summary inclusion **[21]**.

2.1.2 Maximum Entropy (add formulas)

Osborne in his work discuss, the problem of extracting sentences for document summarising is covered in the research. In order to construct a summary, sentence extraction involves selecting important sentences from a document. For this purpose, the authors investigate the usage of naive Bayes classifiers and maximum entropy **[22]**.By maximising the weights given to various factors, the maximum entropy classifier enables the integration of multiple knowledge sources. It may efficiently overlook aspects that aren't important by adapting to them. The naïve Bayes classifier, in contrast, treats features categorically and assumes independence between them **[22]**.The research provides a maximum entropy classifier-based incremental approach to sentence extraction. Based on attributes taken from the sentence and its context, the classifier decides whether or not to extract a sentence.The authors acknowledge that the maximum entropy approach has limitations, particularly when using standard features on technical documents. The features that predict sentence extraction tend to be specific and infrequent, leading to low recall. However, the addition of an optimized prior to the maximum entropy classifier improves its performance compared to naive Bayes **[22]**.The importance of considering lexical cohesion factors in sentence extraction, such as lexical repetition, ellipsis, and co-reference. These factors contribute to the overall discourse structure and need to be modeled in an ideal sentence extraction system.The general framework for sentence extraction using maximum entropy modeling and presents a comparison with the naive Bayes classifier.

2.1.3 Text Rank Algorithm

TextRank Algorithm is a graph based method for text processing. It is unsupervised method for keyword and sentence extraction from single document.As we know graph are used to represent data as vertex and edges as relationship between them.In textRank sentence is treated as vertex, “voteing” or “recommendation” is treated as edge between two sentences in a graph.Consider a directed graph $G = (V, E)$ consisting of a set of vertices V and a set of edges E, where the edges are a subset of $V \times V$. For a given vertex V_i, $In(V_i)$ represents the set of vertices that point to it (predecessors), and $Out(V_i)$ represents the set of vertices that V_i points to (successors). The score of a vertex V_i is defined as follows:

$$S(Vi) = (1 - d) + d \times \sum_{j \in In(Vi)} \frac{1}{|Out(Vj)|} S(Vj)$$

Where d is a damping factor that can be range from 0 to 1. Where $S(i)$ is the score of Vi **[23][4]**.

Depending on the application of the algorithms, text unit of different length can be added as vertices in the graph e.g. words,collcation,entire sentence or other like wise, it is the application that define the type of relations that are used to make connections between any two such vertices, For Example lexical or semantic relation, contextual overlap,etc.

We require an order to allow the use of graph-based ranking algorithms for natural language texts. Creation of a graph that depicts the text and connections between words or other text entities in a meaningful way.Words, collocations, complete sentences, and other text units can be added as vertices in the graph depending on the specific application available. Similar to how the type of relations used to draw connections between any two of these vertices is determined by the application **[23][4]**.

The process of applying graph-based ranking algorithms to natural language texts involves the following fundamental phase, regardless of the kind and characteristics of the individual elements added to the graph.

1. Select the text components that best represent the given task, and incorporate them into the graph as individual vertices..
2. Establish connections between the vertices of the graph by identifying the relationships that link these textual elements. These connections can be represented by edges, which can have different characteristics such as weights or directions.

3. The graph-based ranking method should be iterated till convergence.
4. Arrange the vertices based on their final scores. Utilize the associated values of each vertex to make decisions regarding ranking and selection.

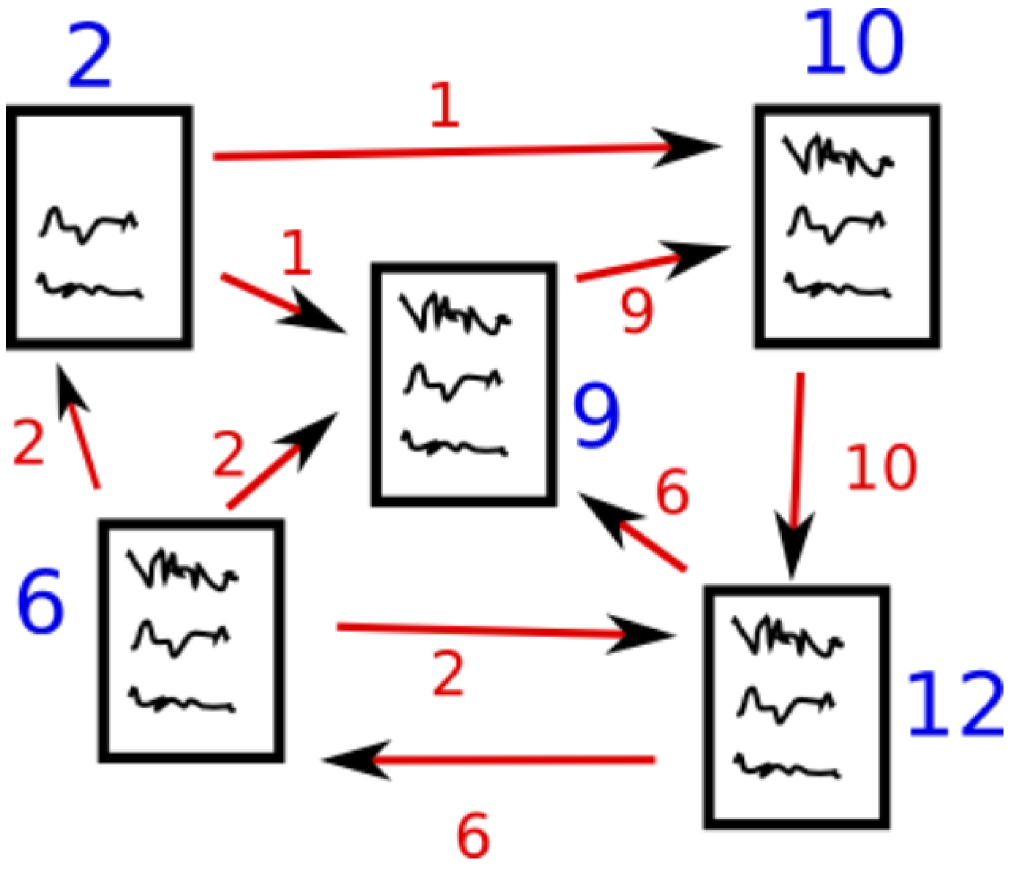


Fig 2.4 TextRank Graph Representation

Applications of Text Rank Algorithm
Keyword Extraction:
A keyword extraction objective is to automatically find the words that best characterize a text in a given text. Such keywords may serve as helpful entries for creating a document collection's automated index, can be utilized to categorize a text,

It might act as a brief summary of the information presented. The implementation of a system for the automatic identification of important terms in a text can also help with the issue of terminology extraction and the development of domain-specific dictionaries. The simplest technique for identifying the "important" phrases in a text may be to use a frequency factor. However, it was demonstrated that this method frequently led to disappointing results. Text Rank instead extracts keywords based on part of speech tags, ranks them based on the co-occurrences of words determined by a sliding window, and then selects potential keywords. **[23][4]**.

Text Rank is expected to produce a set of phrases or words that best describe a particular natural language text. The ranking units are Consequently, sequences of one or more lexical units that are retrieved from text and serve as the vertices for the text network. Any link (edge) that may be added between two of these vertices and that can be formed between two lexical units is potentially beneficial.It makes use of a co-occurrence connection that is influenced by the separation between word occurrences. Two vertices are connected if their corresponding lexical units co-occur within a window of maximum words, which can be set anywhere between 2 and 10 words. Similar to the semantic linkages that have been proven effective for the process of word meaning disambiguation, co-occurrence links describe relationships between syntactic components and serve as cohesion indicators for a specific text.Here illustrates how the unsupervised TextRank keyword extraction algorithm works. Tokenizing the text and adding part-of-speech tags are the first steps needed to enable the use of syntactic filters. We only consider single words as candidates for addition to the graph because adding all combinations of sequences made up of multiple lexical units (n-grams) would result in an excessive increase in the graph's size. Instead, multi-word keywords are eventually reconstructed during the post-processing stage **[23][4]**.

Then, all lexical units that pass the syntactic filter are added to the graph, and an edge is generated between each lexical unit that co-occurs within a window of words. After being created, an undirected unweighted graph is iterated over a certain number of times until it converges, usually between 20 and 30 times at a limit of 0.0001.
After assigning a final score to each vertex in the graph, the vertices are ranked in reverse order of their final scores, with the top vertices being retained for further analysis. While any set value may be selected, a more flexible technique that relies the number of keywords on the length of the text is frequently used, typically between 5 and 20 keywords One-third of the data used in our trials, which consists of short summaries, is set as the network's number of vertices.
<u>Sentence Extraction:</u>
Since both applications aim to identify sequences that are more "representative" of the provided text, the challenge of sentence extraction may

be related to that of keyword extraction.In keyword extraction, the candidate text units are constructed from words or phrases, but we deal with complete sentences in sentence extraction. It turns out that TextRank is a good fit for these kinds of applications since it enables recursively computing rankings across text units using data from the complete text.

To use TextRank, we must first create a text-related graph in which the vertices represent the units that will be scored. A vertex is added to the graph for each sentence in the text as the objective of sentence extraction is to rate full sentences **[23][4]**.

Given that the text units being analyzed are larger in size and the concept of co-occurrence is not applicable in such extensive contexts, the conventional co-occurrence relation used for keyword extraction cannot be employed here. Instead, we are introducing a new relation that establishes a connection between two phrases based on their degree of content overlap, which can be considered as a measure of "similarity." This relationship between two sentences can be likened to a process of "recommendation," where a sentence addressing specific concepts in the text suggests that the reader refers to other sentences discussing the same concepts. Consequently, a connection can be established between any two sentences that share common information. The degree of overlap between sentences can be computed by counting the shared tokens across their lexical representations. Alternatively, syntactic filters can be applied to consider only words belonging to specific syntactic categories, such as nouns, verbs, or all open-class words. Additionally, a normalization factor is utilized by dividing the content overlap of two phrases by the length of each sentence. This prevents favoring longer sentences in the similarity calculation. Formally, when given two sentences S_i and S_j, where a sentence is represented by the set of N_i words appearing in the sentence:
$S_i = w_1^i.\ w_2^i.\ ....\ w_{N_i}^i$, the similarity between S_i and S_j is defined as:

$$Similarity(S_i, S_j) = \frac{\left|\{w_k \mid w_k \in S_i \,\&\, w_k \in S_j\}\right|}{log(|S_i|) + log(|S_j|)}$$

We are now assessing the effects of other word similarity metrics on the impact of summarization, including string kernels, cosine similarity, longest common subsequence, etc **[23][4]**.

2.1.4 Artificial Neural Network(table and formulas)

NetSum is an innovative approach for automatically summarising papers that makes use of neural networks to extract essential data. Each sentence is given a set of attributes that the system uses to determine its importance in the document. These properties, which provide useful context information, are based on Wikipedia entities and news search query logs.The RankNet learning algorithm is used to train the system. It is a pair-based sentence ranker that gives each phrase in the text a score to show how important they are. The training process involves labeling a training set and extracting features from the sentences.The system then creates a ranked list of sentences for each text after learning the distribution of properties that correspond to the best sentences. Documents gathered from CNN.com, each of which contains highlights and an article, are used to evaluate NetSum. The ROUGE-1 metric, which evaluates the calibre of the generated summaries, is used to gauge the success of the system. On more than 70% of the document collection, NetSum outperforms the typical baseline in the ROUGE-1 metric, demonstrating its better summarising skills. The technique of automatically condensing papers into shorter forms, known as summarization, has been the focus of research for many years. Early approaches focused on linguistic and statistical methods to identify key phrases and concepts in sentences or across multiple documents. More recently, machine learning techniques have been successfully applied to summarization, including binary classifiers, Markov models, Bayesian methods, and heuristic approaches **[24]**.

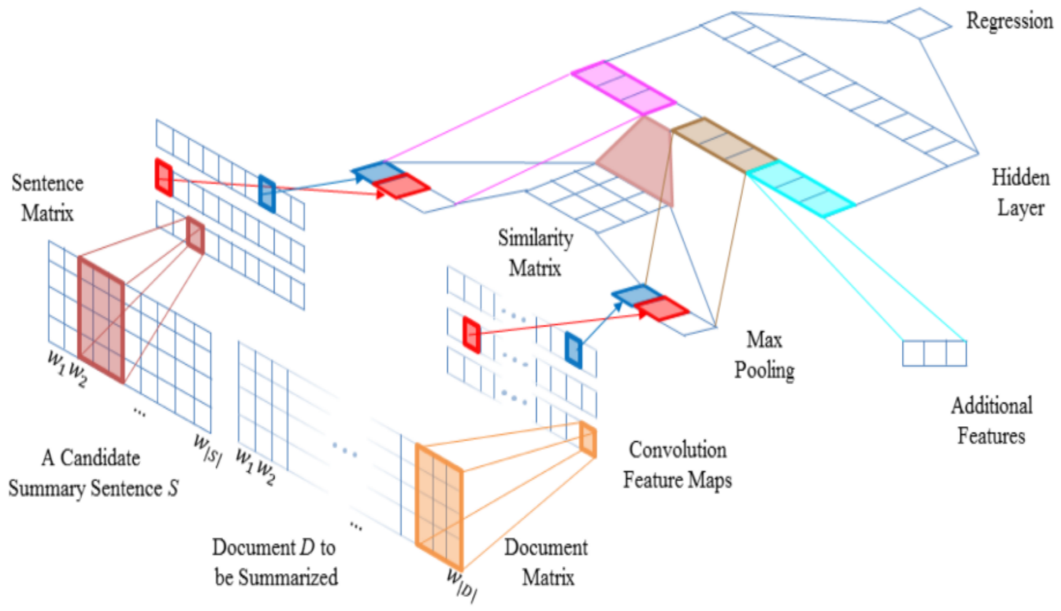


Fig 2.5 Artificial Neural Network

The decline in studies on single-document summarization is attributed to the discontinuation of the Document Understanding Conference (DUC) task and the perception that single-document summarization is more challenging than multi-document summarization. However, with the increasing availability of information on the internet, single-document summarization remains crucial for efficient information access **[24]**.

The motivation for NetSum comes from the addition of story highlights on CNN.com, which allows readers to quickly gather information about news articles. These highlights provide a brief overview of the article in bullet points, making them easily scannable. Despite the introduction of highlights and previous summarization systems, none have surpassed the strong baseline of choosing the first few sentences as the summary.NetSum addresses this challenge by using a neural network ranking algorithm and incorporating third-party datasets to enhance sentence features. By leveraging the power of neural networks, NetSum outperforms the baseline with statistical significance, providing more accurate and informative summaries.The system is designed for single-document extract summaries of newswire articles. It focuses on extracting three sentences that best match the highlights generated by humans. Two tasks are defined: creating a summary block that matches the highlights as a whole and preserving the ordering of sentences to match individual highlights.The evaluation corpus for NetSum consists of 365 news documents collected from CNN.com. Each document includes the title, timestamp, story highlights, and article text. The dataset provides a diverse range of news articles for training and evaluation purposes **[24]**.

In conclusion, NetSum introduces a novel approach to automatic summarization using neural networks. By extracting features from sentences and training a ranker, NetSum identifies the most important sentences in a document. The system outperforms the standard baseline on the ROUGE-1 measure, demonstrating its effectiveness in generating high-quality summaries. With the increasing need for efficient access to large amounts of information, single-document summarization remains a crucial area of research, and NetSum offers a promising solution.

2.1.5 BERT for extractive summarization

For extractive text summarization, the BERT (Bidirectional Encoder Representations from Transformers) language model is an efficient tool. The process of extractive summarising is choosing and combining the key words or sentences from a source material to produce a summary. BERT might be customised expressly for this kind of task due to its capacity to gather contextual information and semantic connections in text. **[25]**.

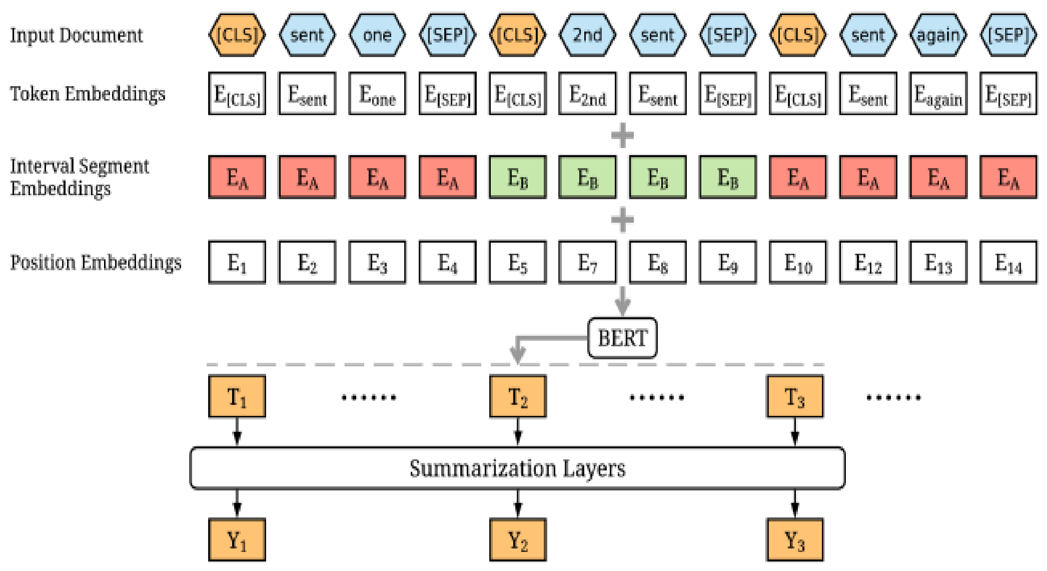
To elaborate on BERT for extractive summarization, let's go through the key steps involved:

1. Pretraining BERT:
   With the use of a disguised language modelling target, BERT is pretrained on a sizable corpus of text data. BERT gains the ability to anticipate masked words in a phrase based on the surrounding context during this pretraining phase. As a result, it is able to capture word and phrase context, which is the basis for its language comprehension skills. **[26]**.
2. Fine-tuning BERT for summarization:
   After pretraining, BERT is fine-tuned on a specific dataset for the task of extractive summarization. The dataset consists of pairs of source documents and their corresponding human-generated summaries. The objective is to teach BERT how to identify the most important sentences or phrases in the document for summarization.
   a. Input Representation:
      To prepare the input for fine-tuning, each sentence or phrase in the document is combined into a single input sequence. Special tokens like [CLS] and [SEP] are inserted to mark the beginning and separation of the segments. Segment embeddings are assigned to tokens to indicate their respective sentence or phrase segments **[25][26]**.
   b. Binary Classification:
      During fine-tuning, BERT is trained as a binary classifier to predict whether a sentence or phrase should be included in the summary or not. Each input sequence is labeled with binary values (0 or 1) to indicate whether it should be part of the summary. This allows BERT to learn to identify the most salient and informative parts of the document **[25][26]**.
   c. Training Objective:
      The fine-tuning process involves optimizing BERT's parameters to minimize the discrepancy between the predicted binary labels and the ground truth labels. This is typically done using techniques like binary cross-entropy loss or sigmoid activation function. The objective is to guide BERT to accurately identify the sentences or phrases that should be included in the summary.
3. Sentence Ranking and Selection

Once BERT is fine-tuned, it can be used for extractive summarization by ranking and selecting the most important sentences or phrases from the document. This can be done using various strategies:

   a. Sentence Similarity:BERT can compute the similarity between each sentence or phrase in the document and a predefined summary length. The sentences or phrases with the highest similarity scores are selected as candidates for the summary. This approach leverages BERT's contextual understanding to identify the most relevant content **[27]**.
   b. Sentence Importance:
   BERT can also assign importance scores to each sentence or phrase based on their predicted probabilities. The sentences or phrases with the highest importance scores are included in the summary. This approach allows BERT to prioritize the most informative and salient parts of the document **[27]**.

4. Summary Generation:
The selected sentences or phrases are combined to create the final summary. Additional post-processing techniques, such as removing redundant information or adjusting the summary length, can be applied to improve the coherence and readability of the generated summary.



Input Document
[CLS]
sent
one
[SEP]
[CLS]
2nd
sent
[SEP]
[CLS]
sent
again
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{sent} E_{one} $E_{[SEP]}$ $E_{[CLS]}$ E_{2nd} E_{sent} $E_{[SEP]}$ $E_{[CLS]}$ E_{sent} E_{again} $E_{[SEP]}$
Interval Segment Embeddings
E_A E_A E_A E_A E_B E_B E_B E_B E_A E_A E_A E_A
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_7 E_8 E_9 E_{10} E_{12} E_{13} E_{14}
BERT
T_1 T_2 T_3
Summarization Layers
Y_1 Y_2 Y_3

Fig 2.6 BERT for Extractive Summarization

BERT for extractive summarization allows for the automatic selection of important sentences or phrases, eliminating the need for manual summarization. It leverages BERT's language understanding capabilities to identify the most relevant content from the document, providing a concise and informative summary **[25][26][27]**.

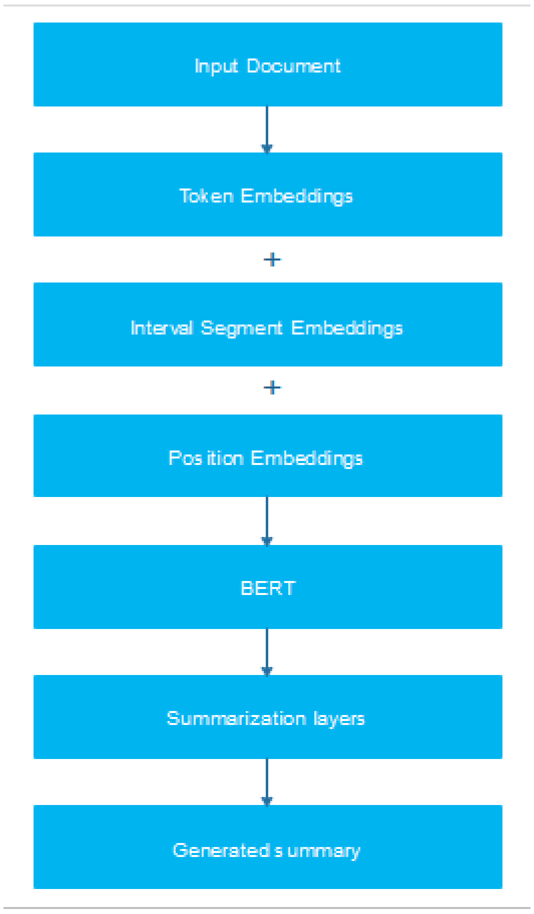


Fig 2.7 Steps involving Bert

It's important to note that while BERT can generate high-quality extractive summaries, it may not capture the entire context or provide a coherent narrative as in abstractive summarization. However, it offers a more data-driven and objective approach to summarization by directly extracting

2.2 Abstractive Text Summarization

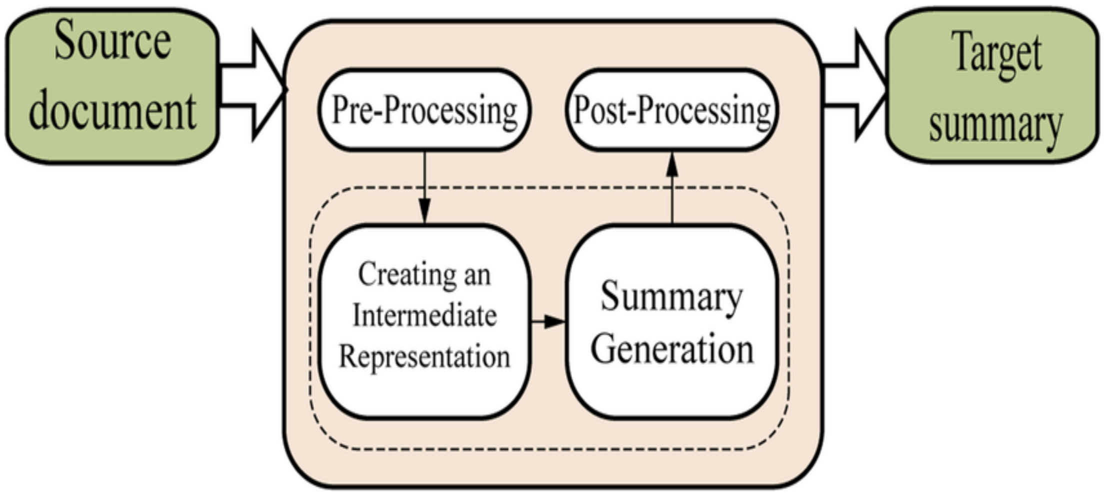


Fig 2.8 Abstractive Text Summarization

Abstractive summarization is a text summarization technique that aims to generate concise and coherent summaries by understanding the content of the source document and rephrasing it in a new way. Unlike extractive summarization, which selects and combines existing sentences or phrases, abstractive summarization goes beyond the source material to create summaries that may include novel sentences and paraphrases. Abstractive summarization involves several key steps and techniques **[28][29]**:

1. **Input Representation**:
   To perform abstractive summarization, the source document is tokenized into individual words or subword units. Each word or subword is then encoded into a numerical representation suitable for processing by the model. Common encoding techniques include word embeddings, such as Word2Vec or GloVe, or subword embeddings like Byte Pair Encoding (BPE) or SentencePiece **[29][30]**.
2. **Sequence-to-Sequence Models**:
   The learning problem of sequence-to-sequence (Seq2Seq) learning is frequently used to describe abstractive summarization. An encoder and a decoder are components of Seq2Seq models. The encoder analyses the original material, extracts the contextual and semantic information, and encodes it into a fixed-length representation. After decoding the encoded form, the decoder then produces the summary. **[30][31]**.

3. **Attention Mechanism**:
   To improve the information flow between the encoder and the decoder, attention mechanisms are commonly employed. Attention allows the model to focus on different parts of the source document while generating the summary. It helps the model align important words or phrases in the document with the corresponding words in the summary, facilitating the generation of coherent and meaningful summaries**[29][30]**.
4. **Training with Ground Truth Summaries**:
   During the training phase, the model is provided with pairs of source documents and their corresponding human-generated summaries. The model's parameters are optimized to minimize the discrepancy between the generated summaries and the ground truth summaries. Techniques like cross-entropy loss or maximum likelihood estimation are used as training objectives.
5. **Beam Search or Sampling**:
   When it comes to generating the actual summaries, decoding techniques such as beam search or sampling are employed. Beam search explores multiple candidate sequences by considering the most likely words at each step, while sampling selects words probabilistically. These techniques help to balance between generating fluent summaries and exploring diverse possibilities.
6. **Copy Mechanism**:
   In abstractive summarization, the model sometimes needs to copy content directly from the source document to ensure accuracy. To address this, copy mechanisms are introduced, allowing the model to selectively copy words or phrases from the source document instead of relying solely on the decoder's generation. This helps to preserve specific details and maintain fidelity to the original text.
7. **Fine-tuning with Reinforcement Learning**:
   To further improve the generated summaries, reinforcement learning techniques can be applied. Reinforcement learning uses reward signals to guide the model towards generating high-quality summaries. It encourages the model to explore different wordings and sentence structures, leading to more creative and diverse summaries.+
8. **Evaluation Metrics**:
   Abstractive summaries are evaluated for quality using a variety of parameters. ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which assesses the degree to which the produced summary

and the ground truth summary coincide in terms of n-gram matches, is one example of a common metric. Aside from these measures, METEOR (Metric for Evaluation of Translation with Explicit ORdering) and BLEU (Bilingual Evaluation Understudy) can also be employed. **[8][9][10]**.

Abstractive summarization offers the advantage of generating more concise summaries that capture the essence of the source document while providing a more coherent narrative. However, it is a challenging task as it requires the model to understand the semantics, context, and nuances of the text and generate grammatically correct and contextually appropriate summaries.

Furthermore, abstractive summarization models often struggle with generating summaries that are factually accurate or consistent. Ensuring the faithfulness and coherence of the generated summaries remains

2.2.1 Encoder-Decoder Models

Encoder-decoder architectures are widely used in text summarization, particularly in the context of abstractive summarization. This approach involves two main components: an encoder and a decoder. The encoder processes the input text, while the decoder generates a concise summary based on the encoded representation. This method has demonstrated impressive results in generating human-like summaries that capture the essence of the source document.

Let's delve deeper into the workings of an encoder-decoder model for text summarization:

1. Encoder:
   The encoder receives the input document, which consists of a sequence of words or tokens. It encodes the document by processing it sequentially. Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) networks, are commonly used for this purpose. The encoder updates its hidden state at each step, incorporating information about the current word and the previous hidden state. By the end of the encoding process, the encoder produces a fixed-length representation that encodes the document's semantics and contextual information **[31][32][33]**.

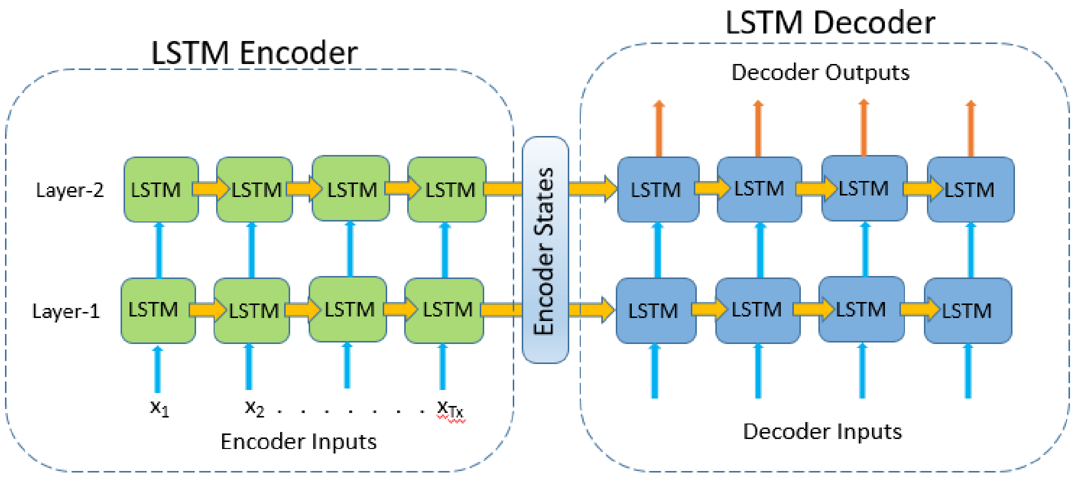


Fig 2.9 LSTM encoder-decoder

2. Attention Mechanism:
   To improve the performance of the encoder-decoder model, an attention mechanism is often employed. The attention mechanism allows the decoder to focus on different parts of the encoded representation during summary generation. It assigns weights or probabilities to different positions in the encoded representation, indicating their relevance or importance for the current decoding step. By considering these weights, the decoder can selectively attend to the most informative parts of the document representation, leading to more accurate and contextually relevant summaries **[31][33]**.
3. Decoder:
   The decoder takes the encoded representation generated by the encoder and generates the summary. Like the encoder, the decoder is typically implemented as an RNN, such as an LSTM or GRU network. During decoding, the decoder maintains its own hidden state, which is updated based on the previously generated words and the attention-weighted combination of the encoded representation. Using this hidden state, the decoder predicts the next word in the summary. This process continues until an end-of-sentence token is generated or a predefined length for the summary is reached **[31][32][33][24]**.
4. Training:
   The encoder-decoder model is taught during the training phase to use an appropriate loss function, frequently cross-entropy loss, to minimise the difference between the produced summary and the reference summary. The parameters of both the encoder and decoder, including word embeddings, RNN weights, and attention weights, are updated

using the backpropagation algorithm through time. This iterative process allows the model to learn how to generate high-quality summaries that align with human-generated summaries **[31][32]**.

5. Training:
   The encoder-decoder model is taught during the training phase to use an appropriate loss function, frequently cross-entropy loss, to minimise the difference between the produced summary and the reference summary. The backpropagation technique is used to update the encoder and decoder's parameters over time, including word embeddings, RNN weights, and attention weights. This iterative procedure enables the model to develop high-quality summaries that are compatible with summaries produced by humans.

Encoder-decoder models have demonstrated their efficacy in text summarization tasks. They have the advantage of being able to generate abstractive summaries, which are not restricted to extracting sentences from the input document. These models can generate summaries that incorporate new phrases, rephrase the information, and provide a more concise and coherent representation. Researchers continue to explore and improve upon the encoder-decoder architecture for text summarization, incorporating techniques such as pointer networks, coverage mechanisms, or reinforcement learning to enhance the quality and fluency of the generated summaries **[32][31][33][34]**.

2.2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language model that has been successfully applied to various natural language processing tasks, including abstractive text summarization. Abstractive summarization involves generating concise and coherent summaries that capture the key information from a given document. BERT, with its ability to understand the context and capture the semantic relationships within text, can be fine-tuned specifically for this task **[25][26][27]**.

To elaborate on BERT for abstractive summarization, let's go through the key steps involved:

1. **Pretraining BERT**:
   BERT is pretrained on a large corpus of text data using a masked language modeling objective. During this pretraining phase, BERT

learns to predict masked words in a sentence based on the surrounding context. By doing so, it captures the contextual representations of words and sentences, which forms the foundation of its language understanding capabilities **[25]**.

2. **Fine-tuning BERT for summarization**:
   After pretraining, BERT is fine-tuned on a specific dataset for the task of abstractive summarization. The dataset consists of pairs of source documents and their corresponding human-generated summaries. The objective is to teach BERT how to generate accurate and informative summaries **[25][26]**.
   a. **Input Representation:**
      To prepare the input for fine-tuning, the source document and the summary are combined into a single input sequence. Special tokens like [CLS] and [SEP] are inserted to mark the beginning and separation of the two segments. Additionally, segment embeddings are assigned to tokens to indicate whether they belong to the document or summary segment.
   b. **Masked Language Model (MLM)**:
      During fine-tuning, BERT is further trained using the masked language modeling objective. In this step, some tokens in the input sequence are randomly masked, and BERT learns to predict the original words based on the surrounding context. By doing so, BERT gains an understanding of the relationships between words and phrases, which is crucial for generating accurate summaries.
   c. **Next Sentence Prediction (NSP)**:
      BERT also undergoes next sentence prediction training. It is presented with pairs of source document-summary sequences, and it learns to predict whether the summary is the actual next sentence following the document. This helps BERT capture the coherence and flow between the document and summary, which is important for generating coherent summaries.
   d. **Training Objective**
      The fine-tuning process involves optimizing BERT's parameters to minimize the discrepancy between the predicted summary and the ground truth summary. This is typically done using techniques like cross-entropy loss or maximum likelihood estimation. The objective is to guide BERT to generate summaries that closely match the human-generated ones.

3. **Decoding the Summary**:
   During inference, BERT is used to generate the summary by decoding the most probable sequence of words. Decoding techniques such as beam search or sampling are employed to explore multiple possible word sequences and select the most appropriate one.
   a. **Beam Search**:
      Beam search is a popular decoding technique that explores multiple candidate word sequences. It maintains a set of top-k candidate sequences and expands them by predicting the next word at each step. The candidates are then ranked based on their probabilities, and the process continues until a maximum length or stopping condition is reached. The generated summary is selected from the candidate pool.
   b. **Sampling**:
      Sampling involves randomly selecting words based on their probabilities. This technique allows for more diversity in the generated summaries but may lead to less optimal or coherent results. Different sampling strategies, such as temperature-based sampling or nucleus sampling, can be used to control the randomness and quality of the generated summaries. **[25][26][27]**

By fine-tuning BERT for abstractive summarization, the model learns to capture the contextual relationships between words and generate concise and coherent summaries that convey the essential information from the source document. However, BERT's conservative nature may result in summaries that closely match

2.3 Evaluation metrics

Evaluating the quality of text summarization systems is crucial to assess their performance and compare different approaches. Various evaluation metrics have been developed to measure the effectiveness of summaries generated by these systems. Here are some commonly used text summarization evaluation metrics:

1. ROUGE (Recall-Oriented Understudy for Gisting Evaluation)
2. BLEU (Bilingual Evaluation Understudy)
3. METEOR (Metric for Evaluation of Translation with Explicit Ordering)
4. CIDEr (Consensus-based Image Description Evaluation)
5. Pyramid

6. Length-Based Metrics
7. Semantic Similarity Metrics

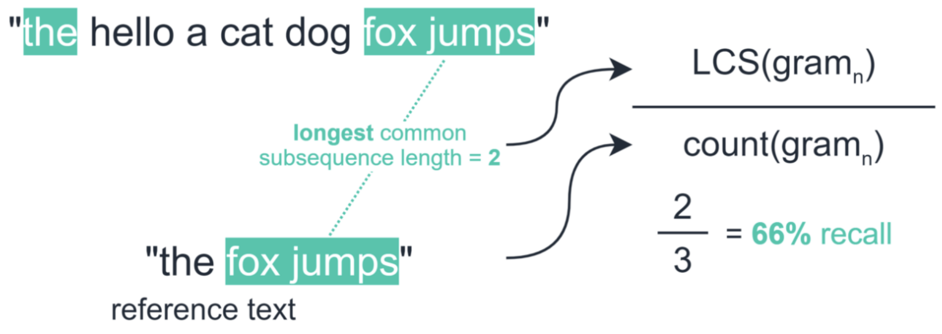
2.3.1 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE is a commonly employed evaluation metric in the field of text summarization. It quantifies the degree of overlap between the summary produced by a system and the reference summary created by humans, based on matching n-grams. ROUGE measures the recall of the generated summary by examining its ability to capture essential information from the reference summary. **[15]**.

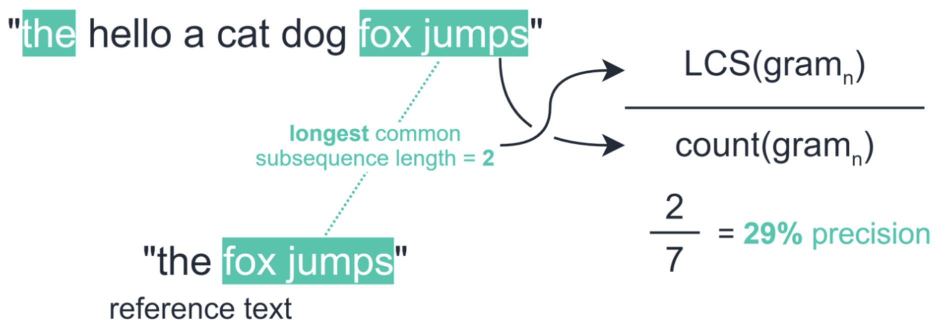
The ROUGE metric consists of several variations, including ROUGE-N, ROUGE-L, and ROUGE-S.

1. ROUGE-N: ROUGE-N measures the n-gram overlap between the generated summary and the reference summary. It considers different values of n, typically ranging from 1 to 4, to evaluate unigram, bigram, trigram, and quadgram matches. ROUGE-N counts the number of n-gram matches between the generated and reference summaries, and then calculates precision, recall, and F-score based on these counts. Precision measures the ratio of matched n-grams in the generated summary to the total number of n-grams in the generated summary, while recall measures the ratio of matched n-grams to the total number of n-grams in the reference summary **[15]**.
2. The longest common subsequence (LCS) between the produced summary and the reference summary is measured by ROUGE-L. The longest set of words to appear in both summaries in the same order is LCS. ROUGE-L measures how well the produced summary can extract significant quotes or passages of text from the reference summary. Based on the length of the LCS and the lengths of the produced and reference summaries, it determines accuracy, recall, and F-score **[15]**.
3. ROUGE-S: ROUGE-S evaluates skip-bigram matches, considering the order of words in the summaries. Skip-bigrams are pairs of words that are not necessarily adjacent but appear in the same order in both summaries. ROUGE-S captures the syntactic similarity between the generated and reference summaries, measuring their ability to maintain the overall sentence structure and word order. Similar to other ROUGE variants, ROUGE-S calculates precision, recall, and F-score based on the number of skip-bigram matches **[15]**

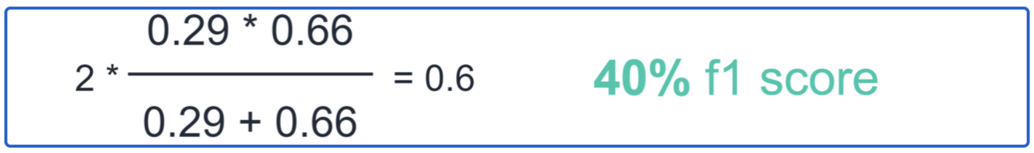
The ROUGE scores range from 0 to 1, with higher scores indicating better quality summaries. ROUGE evaluates the content overlap between the generated summary and the reference summary, providing a quantitative measure of summary quality. It has been widely used in research and benchmarking to compare different summarization systems and approaches.However, it is important to note that ROUGE has its limitations. It does not capture the overall coherence, fluency, or readability of summaries, as it focuses primarily on content overlap. ROUGE metrics also do not account for paraphrasing or rephrasing of information in the summaries. Additionally, the choice of n-gram order and the specific ROUGE variant can impact the results, and different ROUGE variants may provide different insights into the quality of the summaries **[15]** .



Our LCS recall calculation



Precision is much the same but we switch our total n-gram count from the reference to the model



And finally, we calculate the F1 score just like we did before

To complement ROUGE evaluation, it is often recommended to combine it with other metrics that capture different aspects of summary quality, such as human evaluation or semantic similarity metrics. This multi-faceted evaluation approach provides a more comprehensive understanding of the strengths and weaknesses of the summarization systems **[15]**.

2.3.2 BLEU (Bilingual Evaluation Understudy)

Natural language processing activities like text summarization and machine translation frequently employ the evaluation metric known as BLEU. It evaluates generated text's quality by contrasting it with one or more reference texts, which are often written by actual experts. The goal of BLEU is to identify the overlap of n-grams (contiguous word sequences) between the reference text and the output that is created.
The BLEU metric computes precision scores for different n-gram orders (unigrams, bigrams, trigrams, etc.) and combines them using a geometric mean. The precision score measures how many of the n-grams in the generated output match with the reference text. Higher precision scores indicate better quality output **[16]**.

Here's how BLEU works in more detail:

1. N-gram Matching:
   BLEU computes the precision for different n-gram orders (typically up to 4-grams) between the generated output and the reference text. It counts the number of overlapping n-grams between the two texts **[16]**.
2. Modified Precision Calculation:
   To avoid favoring overly short outputs, BLEU includes a modified precision calculation. It counts the maximum number of times any n-gram appears in any single reference text and limits the count of that n-gram in the generated output to that maximum value. This modification prevents artificially high scores when the generated output simply repeats a few n-grams from the reference **[16]**.
3. Penalty for Short Outputs:
   BLEU penalizes short outputs to discourage generating very few words as summaries. It includes a brevity penalty that reduces the BLEU score for outputs that are significantly shorter than the reference text. This penalty incentivizes generating summaries of appropriate length **[16]**.
4. Cumulative BLEU Score:

The precision scores for different n-gram orders are combined using a geometric mean. This cumulative BLEU score reflects the overall quality of the generated output, considering the match at different levels of granularity (unigrams, bigrams, trigrams, etc.). The geometric mean balances the contribution of different n-gram orders **[16]**.

BLEU scores range from 0 to 1, with 1 indicating a perfect match between the generated output and the reference text. However, it's important to note that BLEU is a reference-based metric and is limited in capturing the semantic quality, coherence, and fluency of the generated output. It primarily focuses on lexical overlap.While BLEU has been widely used, it has some limitations. For instance, it may assign low scores to outputs that differ in phrasing but still convey the same meaning. It also does not account for word order variations or structural differences between the generated output and the reference text. Therefore, BLEU scores should be interpreted with caution and used in combination with other evaluation metrics and human judgment to get a comprehensive assessment of the quality of generated text **[16]**.

Despite its limitations, BLEU provides a quick and automated way to evaluate the quality of generated outputs, enabling researchers and practitioners to compare different systems and track their performance over time. It has become a standard metric for many natural language processing tasks, including machine translation and text summarization **[16]**.

2.3.2 METEOR (Metric for Evaluation of Translation with Explicit Ordering)

METEOR is an evaluation metric commonly used in machine translation and text summarization tasks. It combines precision, recall, and alignment-based measures to assess the quality of generated summaries or translations. METEOR incorporates various linguistic aspects, including unigram matching, stemming, synonymy, and word order, to provide a comprehensive evaluation **[17]**.

To elaborate on METEOR, let's delve into its key components and how they contribute to the metric:

1. Unigram Matching:
   METEOR computes the precision and recall of unigrams (individual words) in the generated summary compared to the reference summary. It takes into account exact matches, stem matches, and matches using WordNet synonyms. This allows the metric to capture semantic similarity beyond strict word-for-word matches **[17]**.
2. Stemming:

METEOR applies stemming to both the generated and reference summaries to account for variations in word forms. By considering the stemmed forms of words, it helps overcome discrepancies arising from inflectional forms or minor variations **[17]**.

3. Synonymy:
   To address synonymy, METEOR uses WordNet, a lexical database, to identify synonyms of words in both the generated and reference summaries. If a word in the generated summary has a synonym that matches a word in the reference summary, it contributes to the matching score. This accounts for lexical variations and promotes flexibility in word choices **[17]**.
4. Word Order:
   METEOR incorporates an alignment-based measure that takes into consideration the word order in the generated and reference summaries. It captures the extent to which the word order in the generated summary resembles that of the reference summary. This aspect is important in evaluating the coherence and fluency of the generated summaries **[17]**.
5. F-score and Harmonic Mean:
   METEOR computes an F-score, which is the harmonic mean of precision and recall. The harmonic mean accounts for cases where precision and recall have a trade-off. It provides a single score that reflects the balance between generating accurate matches and covering the content of the reference summary **[17]**.
6. Implementation Variations:
   Different implementations of METEOR may have slight variations in their specific calculations and parameter settings. These variations can affect the final scores obtained using METEOR. Therefore, it is important to refer to the specific implementation and guidelines provided for consistent and accurate evaluation **[17]**.

METEOR has gained popularity due to its ability to incorporate various linguistic aspects, providing a more comprehensive evaluation compared to simple matching-based metrics like BLEU. It considers not only exact word matches but also accounts for stemming, synonymy, and word order. By capturing these linguistic properties, METEOR aims to align more closely with human judgment and evaluation of summaries or translations.While METEOR is a valuable evaluation metric, it is important to note that no single metric can fully capture the quality and nuances of human-generated summaries. Complementary evaluation methods, such as manual human evaluation or qualitative analysis, can provide additional insights into the overall coherence, readability, and informativeness of the generated summaries **[17]**.

2.3.3 CIDEr (Consensus-based Image Description Evaluation)

CIDEr is an evaluation metric initially developed for image captioning, but it has also been adapted for text summarization evaluation. CIDEr measures the consensus between the generated summary and multiple reference summaries, aiming to capture the quality and effectiveness of the generated summaries.CIDEr incorporates both n-gram precision and term frequency-inverse document frequency (TF-IDF) measures to evaluate the importance of words in the summaries. It considers the consensus among the reference summaries, ensuring that the generated summary aligns well with the human-generated references.The calculation of CIDEr involves several steps. First, the generated summary and the reference summaries are tokenized into individual words or subword units. Then, the n-gram precision is computed by comparing the n-gram counts between the generated summary and the reference summaries. This precision measure assesses how well the generated summary matches the n-grams present in the references.Next, the term frequency-inverse document frequency (TF-IDF) weighting is applied to capture the importance of words. TF-IDF assigns higher weights to words that appear more frequently in the summaries but are less common in the overall dataset. This weighting scheme helps prioritize important words and discourage using generic or frequently occurring terms.
Finally, the n-gram precision and TF-IDF scores are combined using a weighted harmonic mean, where the weights are determined through empirical analysis. The resulting CIDEr score provides an overall measure of the quality and effectiveness of the generated summary, taking into account both the content overlap with the references and the importance of words.CIDEr has gained popularity as an evaluation metric because it considers the consensus among multiple reference summaries, capturing a broader range of valid summary variations. It addresses some limitations of traditional evaluation metrics that solely focus on n-gram matches, such as BLEU or ROUGE. By incorporating TF-IDF weighting, CIDEr can capture the specificity and importance of words in the summaries, providing a more nuanced evaluation.

However, it's worth noting that CIDEr, like other automatic evaluation metrics, may not fully capture all aspects of summary quality, such as coherence, logical flow, or overall coherence. Human evaluation is still crucial for assessing these aspects. Nonetheless, CIDEr serves as a valuable tool in the evaluation toolkit, complementing other metrics to provide a comprehensive assessment of the generated summaries.

2.3.4 Length-based metrics

Length-based metrics in text summarization assess the quality of summaries based on their length compared to the reference summaries. These metrics provide insights into the conciseness and informativeness of the generated summaries. Here are some commonly used length-based metrics:

1. Summary-to-Reference Length Ratio:
   The summary to reference length ratio evaluates the difference between the lengths of the generated and reference summaries. It calculates the degree to which the produced summary deviates from the anticipated length. Ratios that are near to 1 show that the summary and the reference are around the same length, but ratios that are much lower or higher may point to problems with concision or information loss.
2. Compression Ratio:
   The summary-to-reference length ratio evaluates the difference between the lengths of the generated and reference summaries. It calculates the degree to which the produced summary deviates from the anticipated length. Ratios that are near to 1 show that the summary and the reference are around the same length, but ratios that are much lower or higher may point to problems with concision or information loss.
3. Relative Compression:
   The length difference between the source material and the output summary is what is measured by relative compression. It determines the summarising process's percentage decrease. The length of the original material is reduced more when the relative compression is higher, demonstrating the manner in which the summarization method expands information.

Length-based metrics provide a straightforward and quantitative assessment of the conciseness of summaries. However, they have limitations as standalone evaluation metrics. Length alone does not guarantee the quality or informativeness of a summary. It is possible to generate short summaries that lack important details or longer summaries that contain irrelevant or redundant information.

To obtain a comprehensive evaluation, length-based metrics are often used in conjunction with other evaluation measures, such as semantic similarity or content coverage metrics. The combination of length-based metrics with other evaluation techniques helps in capturing both the conciseness and the informativeness of the summaries.

Additionally, it is important to consider the nature of the source documents and the desired summary length when interpreting length-based metrics. Some documents may inherently require longer summaries due to their complexity or the amount of essential information. Different domains or genres may also have different expectations regarding summary length.
In summary, length-based metrics in text summarization provide a quantitative assessment of the conciseness and compression achieved by the summarization process. They are useful for understanding the trade-off between summary length and the preservation of key information. However, they should be used in conjunction with other evaluation measures to ensure a comprehensive evaluation of summary quality.

2.3.5 Semantic similarity metrics

Semantic similarity metrics in text summarization aim to measure the similarity or relatedness between the generated summary and the reference summary based on their semantic content. These metrics go beyond surface-level matching of words or n-grams and consider the meaning and context of the text. Here are some commonly used semantic similarity metrics in text summarization:

1. Word Mover's Distance (WMD):
   WMD calculates the minimal distance that words from one text must "travel" to match terms in the other text in order to determine how different two texts are. To measure the semantic similarity of words, word embeddings are used. WMD takes word context and total meaning into account, enabling a more thorough evaluation of semantic similarity.

2. Smoothed Word Embedding Similarity (ESim):
   ESim uses smoothed word embeddings to determine the semantic similarity between two texts. Between the word vectors in the created summary and the reference summary, it computes the cosine similarity. By smooothing the vectors, ESim solves the sparsity problem of word embeddings and produces a more reliable indicator of semantic similarity.
3. Vector Space Model (VSM):
   The VSM represents documents as vectors in a high-dimensional space. Two texts are compared for similarity based on the cosine similarity between their vector representations. The VSM system captures the semantic relationships between words by taking into consideration the patterns of word co-occurrence in the texts. Semantically, the generated summary may be assessed in connection to the reference summary.

These semantic similarity metrics help evaluate the quality of summaries by considering the meaning and context of the text. They enable a more nuanced understanding of the semantic relationships between words and phrases, capturing the overall coherence and relatedness of the summaries. By utilizing word embeddings or vector representations, these metrics can handle synonyms, paraphrases, and related concepts, enhancing the evaluation of abstractive summarization models.

It is worth noting that semantic similarity metrics are typically used in conjunction with other evaluation metrics, such as ROUGE or BLEU, to provide a comprehensive assessment of summarization quality. While these metrics provide valuable insights, they still have limitations and may not fully capture the complex semantic understanding required for evaluating human-like summarization. Thus, it is often recommended to combine automated metrics with human evaluation to obtain a more comprehensive and accurate assessment of summary quality.

CHAPTER 3 METHODOLOGY

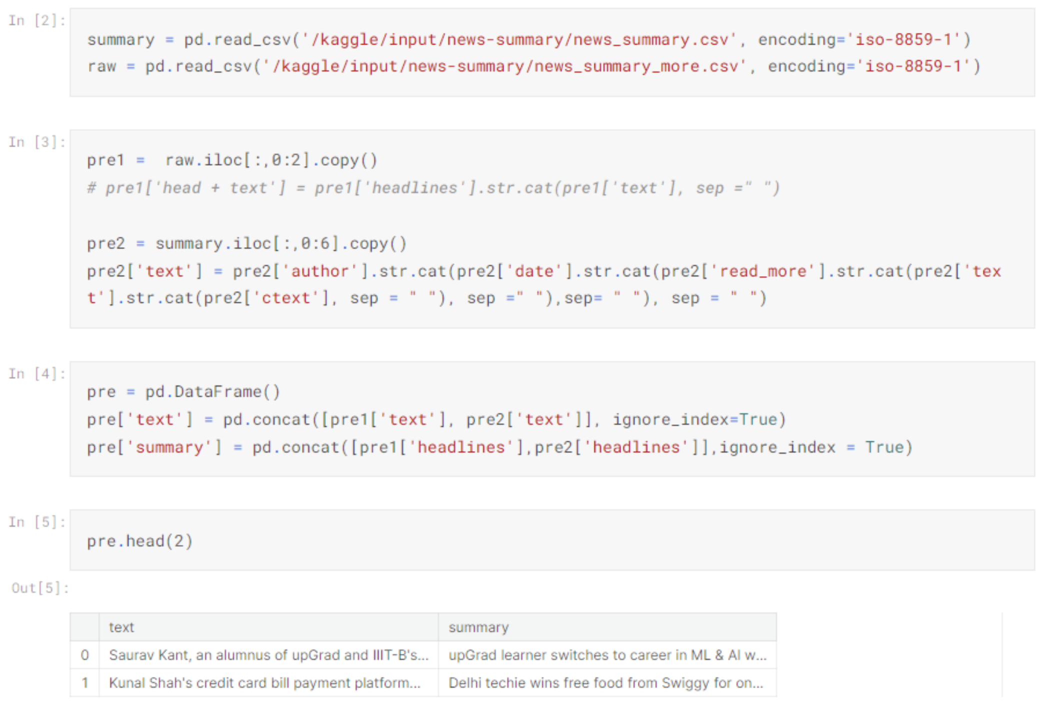
The Following steps are taken in this project:

1. **Data Collection**: In this step, data is collected for **GloVe: Global Vectors for Word Representation** and **News Summary**.
2. **Data Analysis:** In this step, the data is being analyesd on different parameter.
3. **Data Pre-Processing**: In this step, the data is split into two categories,i.e, Training data and Testing data.
4. **Model Building and Training:** In this step, Machine Learning models are created and trained on the dataset processed in the above stages.
5. **Prediction**: The sample dataset is used to predict the outcome of the trained model.
6. **Compare and Evaluate**: In this step, performance of different models is evaluated and compared by different parameters like Precision, Recall, Accuracy, F-score etc.

3.1 Data Collection

GloVe is a widely used dataset and methodology in natural language processing (NLP) that plays a crucial role in text summarization tasks. It is designed to capture semantic and syntactic relationships between words and create vector representations that encode meaningful information about word usage.GloVe gives pre-trained word embeddings, which is a vector representations of words in a continuous vector space. These embeddings are learned from large-scale text corpora, such as Wikipedia or Common Crawl. The key idea behind GloVe is to factorize the word co-occurrence matrix to capture the statistical relationships between words. By doing so, GloVe produces word embeddings that encode the contextual information of words based on their co-occurrence patterns.In the context of text summarization, GloVe embeddings can be leveraged to enhance the understanding of the input text and improve the performance of summarization models. The use of GloVe embeddings allows the model to capture the semantic relationships between words and make more informed decisions when generating summaries.When summarizing a text, the model can utilize the GloVe embeddings to compute the similarity between words or phrases. By measuring the cosine similarity between the word vectors, the model can identify important concepts and select relevant sentences or phrases for inclusion in the summary. This helps to capture the essence of the original text and produce concise summaries that

retain the main points.GloVe embeddings have the advantage of being pre-trained on large-scale datasets, making them useful for tasks where training data is limited. They provide a way to transfer knowledge from the general language domain to the specific task of summarization. By incorporating the GloVe embeddings into the summarization model, it becomes more capable of understanding the underlying semantics of the text and generating coherent and informative summaries.
The dataset used for this project consists of 4,515 instances and includes various fields such as the author's name, headlines, URL of the article, a short text snippet, and the complete article. The summarized news data was collected from Inshorts, while the news articles were scraped from reputable sources such as Hindu, Indian Times, and Guardian. The time period covered in the dataset spans from February to August 2017.



```
In [2]:
summary = pd.read_csv('/kaggle/input/news-summary/news_summary.csv', encoding='iso-8859-1')
raw = pd.read_csv('/kaggle/input/news-summary/news_summary_more.csv', encoding='iso-8859-1')
```

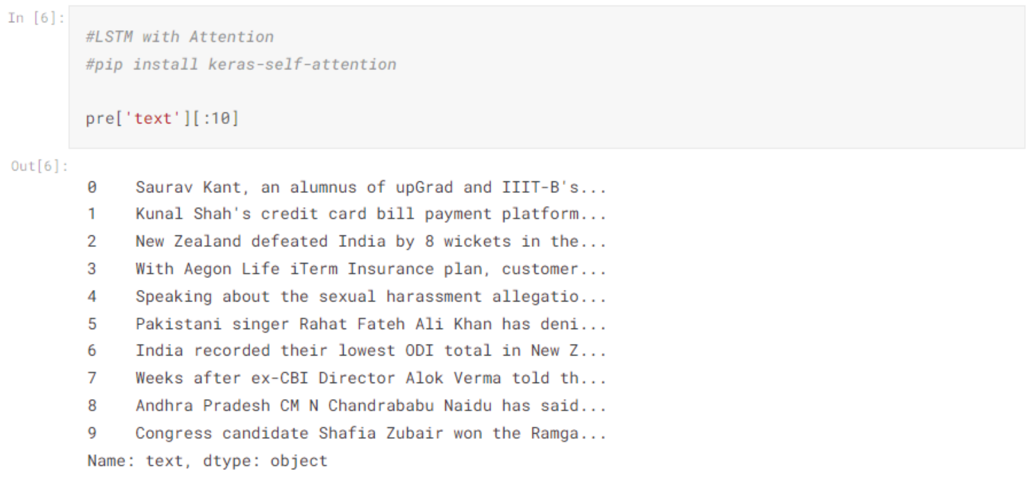
```
In [3]:
pre1 =  raw.iloc[:,0:2].copy()
# pre1['head + text'] = pre1['headlines'].str.cat(pre1['text'], sep =" ")

pre2 = summary.iloc[:,0:6].copy()
pre2['text'] = pre2['author'].str.cat(pre2['date'].str.cat(pre2['read_more'].str.cat(pre2['tex
t'].str.cat(pre2['ctext'], sep = " "), sep =" "),sep= " "), sep = " ")
```

```
In [4]:
pre = pd.DataFrame()
pre['text'] = pd.concat([pre1['text'], pre2['text']], ignore_index=True)
pre['summary'] = pd.concat([pre1['headlines'],pre2['headlines']],ignore_index = True)
```

```
In [5]:
pre.head(2)
```

Out[5]:

	text	summary
0	Saurav Kant, an alumnus of upGrad and IIIT-B's...	upGrad learner switches to career in ML & Al w...
1	Kunal Shah's credit card bill payment platform...	Delhi techie wins free food from Swiggy for on...



```
In [6]:
#LSTM with Attention
#pip install keras-self-attention

pre['text'][:10]
```

```
Out[6]:
0    Saurav Kant, an alumnus of upGrad and IIIT-B's...
1    Kunal Shah's credit card bill payment platform...
2    New Zealand defeated India by 8 wickets in the...
3    With Aegon Life iTerm Insurance plan, customer...
4    Speaking about the sexual harassment allegatio...
5    Pakistani singer Rahat Fateh Ali Khan has deni...
6    India recorded their lowest ODI total in New Z...
7    Weeks after ex-CBI Director Alok Verma told th...
8    Andhra Pradesh CM N Chandrababu Naidu has said...
9    Congress candidate Shafia Zubair won the Ramga...
Name: text, dtype: object
```

3.2 Data Analysis

Data analysis in machine learning is a fundamental process that involves various tasks aimed at understanding, processing, and modeling data to extract valuable insights and make informed decisions. It plays a crucial role in the development of effective machine learning models and ensures the reliability and quality of the results obtained.The process of data analysis typically begins with the collection of relevant data from different sources, such as databases, APIs, or online repositories. This data may be in structured or unstructured formats and needs to be carefully gathered for further analysis.Once the data is collected, the next step is to clean and preprocess it. This involves removing any inconsistencies, errors, or missing values present in the data. Tasks such as handling missing data, removing duplicates, and normalizing data are performed to ensure data quality.Once the data is prepared, the appropriate machine learning model is selected based on the problem at hand. This involves choosing the right algorithm and optimizing hyperparameters. The data is then split into training and testing sets,in the ratio of 70:30 **[35]** and the model is trained on the training data.

```
In [7]:
import re

#Removes non-alphabetic characters:
def text_strip(column):
    for row in column:

        #ORDER OF REGEX IS VERY VERY IMPORTANT!!!!!!

        row=re.sub("(\\t)", ' ', str(row)).lower() #remove escape charecters
        row=re.sub("(\\r)", ' ', str(row)).lower()
        row=re.sub("(\\n)", ' ', str(row)).lower()

        row=re.sub("(__+)", ' ', str(row)).lower()   #remove _ if it occors more than one time consecutively
        row=re.sub("(--+)", ' ', str(row)).lower()   #remove - if it occors more than one time consecutively
        row=re.sub("(~~+)", ' ', str(row)).lower()   #remove ~ if it occors more than one time consecutively
        row=re.sub("(\+\++)", ' ', str(row)).lower()   #remove + if it occors more than one time consecutively
        row=re.sub("(\.\.+)", ' ', str(row)).lower()   #remove . if it occors more than one time consecutively

        row=re.sub(r"[<>()|&©ø\[\]\'\",;?~*!]", ' ', str(row)).lower() #remove <>()|&©ø"',;?~*!

        row=re.sub("(mailto:)", ' ', str(row)).lower() #remove mailto:
        row=re.sub(r"(\\x9\d)", ' ', str(row)).lower() #remove \x9* in text
        row=re.sub("([iI][nN][cC]\d+)", 'INC_NUM', str(row)).lower() #replace INC nums to INC_NUM
        row=re.sub("([cC][mM]\d+)|([cC][hH][gG]\d+)", 'CM_NUM', str(row)).lower() #replace CM# and CHG# to CM_NUM


        row=re.sub("(\.\s+)", ' ', str(row)).lower() #remove full stop at end of words(not between)
        row=re.sub("(\-\s+)", ' ', str(row)).lower() #remove - at end of words(not between)
        row=re.sub("(\:\s+)", ' ', str(row)).lower() #remove : at end of words(not between)

        row=re.sub("(\s+.\s+)", ' ', str(row)).lower() #remove any single charecters hanging between 2 spaces

        #Replace any url as such https://abc.xyz.net/browse/sdf-5327 ====> abc.xyz.net
        try:
            url = re.search(r'((https*:\/*)([^\/\s]+))(.[^\s]+)', str(row))
            repl_url = url.group(3)
            row = re.sub(r'((https*:\/*)([^\/\s]+))(.[^\s]+)',repl_url, str(row))
        except:
            pass #there might be emails with no url in them



        row = re.sub("(\s+)",' ',str(row)).lower() #remove multiple spaces

        #Should always be last
        row=re.sub("(\s+.\s+)", ' ', str(row)).lower() #remove any single charecters hanging between 2 spaces



        yield row
```

3.3 Data Pre-Processing

Machine learning requires the preparation of raw data for analysis and modelling, which is known as data preprocessing. It encompasses several important tasks that enhance the quality and effectiveness of the machine learning algorithm.One crucial aspect of data preprocessing is data cleaning. This involves handling missing data, dealing with outliers, and removing irrelevant or redundant information. Missing data can be addressed by imputing values using techniques like mean, median, or interpolation. Outliers, which are data points that deviate significantly from the norm, can be identified and treated using statistical techniques or domain knowledge.Another important step is data integration, which involves combining data from different sources into a unified dataset. This may require resolving inconsistencies in attribute names, data types, or encoding to ensure compatibility and consistency. The process of transforming data into a format that can be analysed is known as data transformation. This entails converting skewed distributions into more normalised distributions, storing categorical variables, and scaling numerical characteristics to a common range. Techniques like one-hot encoding, label encoding, normalization, and logarithmic transformations are commonly used.Feature selection is a crucial step in data preprocessing, where relevant features are selected to improve model performance and reduce dimensionality. Removing irrelevant or redundant features can enhance the accuracy and training time of the model. The most useful characteristics may be found using methods including correlation analysis, feature importance, and model-based selection.Data preparation also involves dividing the dataset into training, validation, and test sets. The validation set aids in hyperparameter tuning, the test set is used to assess the performance of the final model, and the training set is utilised to train the model.

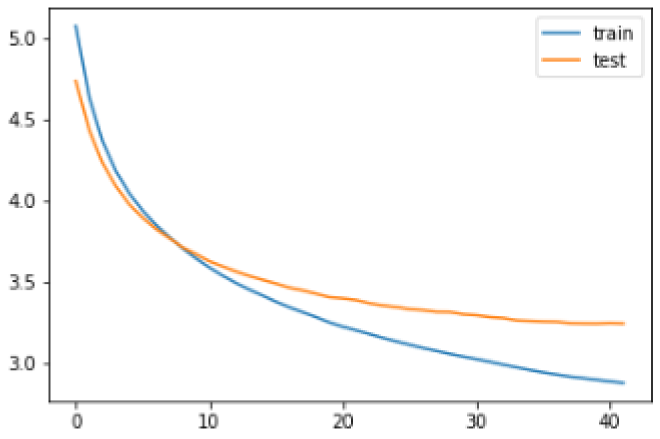
In [35]:

```python
print("Size of vocabulary from the w2v model = {}".format(x_voc))

K.clear_session()

latent_dim = 300
embedding_dim=200

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

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

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

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

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

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

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

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

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

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

model.summary()
```

```
Size of vocabulary from the w2v model = 33412
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 100, 200)     6682400     input_1[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 100, 300), ( 601200      embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 100, 300), ( 721200      lstm[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 200)    2316200     input_2[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 100, 300), ( 721200      lstm_1[0][0]
__________________________________________________________________________________________________
lstm_3 (LSTM)                   [(None, None, 300),  601200      embedding_1[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 11581)  3485881     lstm_3[0][0]
==================================================================================================
Total params: 15,129,281
Trainable params: 15,129,281
Non-trainable params: 0
__________________________________________________________________________________________________
```

3.4 Model Building and Training

The model training process starts after choosing the right model architecture or method. The model taught the underlying patterns and connections between the input characteristics and the target variable using the training set. Optimization algorithms like gradient descent iteratively update the model parameters to minimize errors or maximize performance on the training data.Hyperparameter tuning is an essential step in optimizing the model. Hyperparameters are configuration settings that govern the model's behavior and performance. Examples include learning rate, number of hidden layers, regularization parameters, and batch size. To determine the optimal set of hyperparameters that enhance model performance, methods like grid search, random search, or Bayesian optimisation are used.The validation set is used to assess the model's performance after it has been trained and optimised. Various evaluation metrics like accuracy, precision, recall, F1 score, mean squared error, or area under the curve (AUC) are used to assess the model's effectiveness. Based on the validation results, the model can be further fine-tuned.Once the model is deemed satisfactory, it undergoes final testing using the independent test set. This guarantees a fair assessment of the model's performance on hypothetical data, demonstrating its applicability and performance in the real world.



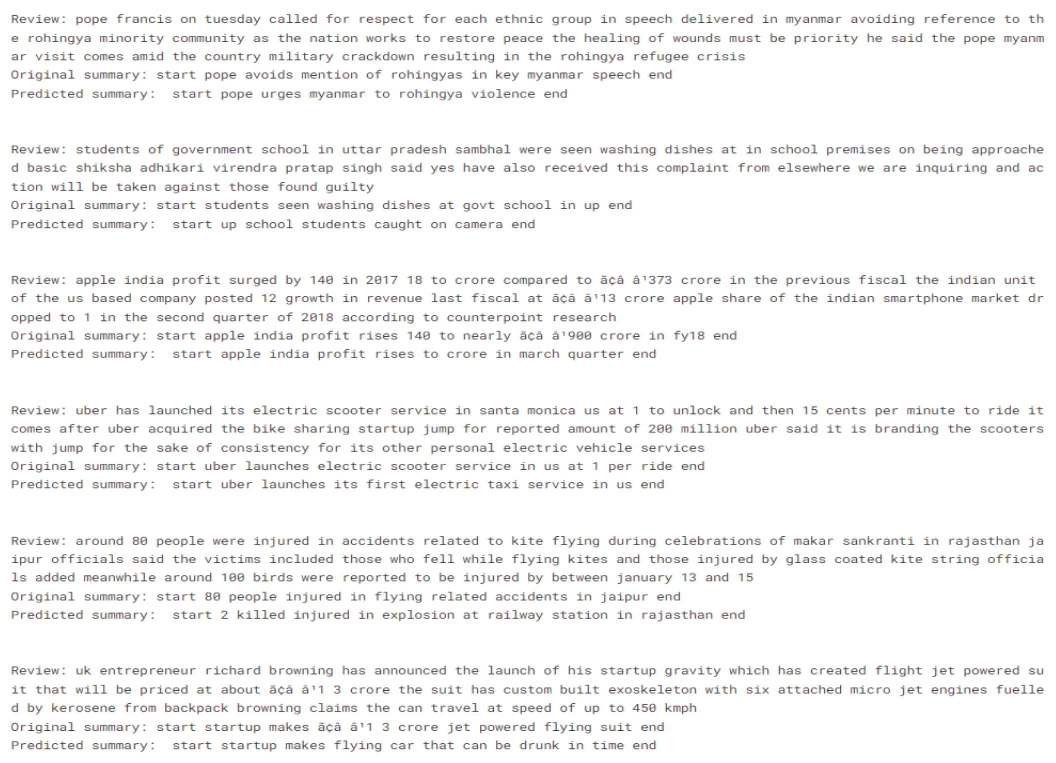
In [38]:

```
history=model.fit([x_tr,y_tr[:,:-1]], y_tr.reshape(y_tr.shape[0],y_tr.shape[1], 1)[:,1:] ,epochs=50,callbacks=[e
s,checkpoint],batch_size=128, validation_data=([x_val,y_val[:,:-1]], y_val.reshape(y_val.shape[0],y_val.shape
[1], 1)[:,1:]))
```

```
Train on 88517 samples, validate on 9836 samples
Epoch 1/50
88517/88517 [==============================] - 417s 5ms/sample - loss: 5.0746 - val_loss: 4.7357
Epoch 2/50
88517/88517 [==============================] - 403s 5ms/sample - loss: 4.6360 - val_loss: 4.4330
Epoch 3/50
88517/88517 [==============================] - 400s 5ms/sample - loss: 4.3639 - val_loss: 4.2293
Epoch 4/50
88517/88517 [==============================] - 398s 4ms/sample - loss: 4.1807 - val_loss: 4.0870
Epoch 5/50
88517/88517 [==============================] - 398s 4ms/sample - loss: 4.0437 - val_loss: 3.9756
Epoch 6/50
88517/88517 [==============================] - 397s 4ms/sample - loss: 3.9372 - val_loss: 3.8931
Epoch 7/50
88517/88517 [==============================] - 396s 4ms/sample - loss: 3.8487 - val_loss: 3.8236
Epoch 8/50
88517/88517 [==============================] - 394s 4ms/sample - loss: 3.7739 - val_loss: 3.7672
Epoch 9/50
88517/88517 [==============================] - 397s 4ms/sample - loss: 3.7047 - val_loss: 3.7102
Epoch 10/50
88517/88517 [==============================] - 395s 4ms/sample - loss: 3.6420 - val_loss: 3.6688
Epoch 11/50
88517/88517 [==============================] - 395s 4ms/sample - loss: 3.5863 - val_loss: 3.6244
Epoch 12/50
88517/88517 [==============================] - 393s 4ms/sample - loss: 3.5354 - val_loss: 3.5929
Epoch 13/50
88517/88517 [==============================] - 395s 4ms/sample - loss: 3.4903 - val_loss: 3.5611
Epoch 14/50
88517/88517 [==============================] - 394s 4ms/sample - loss: 3.4495 - val_loss: 3.5334
Epoch 15/50
88517/88517 [==============================] - 394s 4ms/sample - loss: 3.4124 - val_loss: 3.5096
Epoch 16/50
88517/88517 [==============================] - 392s 4ms/sample - loss: 3.3750 - val_loss: 3.4869
Epoch 17/50
88517/88517 [==============================] - 392s 4ms/sample - loss: 3.3407 - val_loss: 3.4615
Epoch 18/50
88517/88517 [==============================] - 391s 4ms/sample - loss: 3.3100 - val_loss: 3.4447
Epoch 19/50
88517/88517 [==============================] - 393s 4ms/sample - loss: 3.2801 - val_loss: 3.4240
Epoch 20/50
88517/88517 [==============================] - 394s 4ms/sample - loss: 3.2475 - val_loss: 3.4026
Epoch 21/50
88517/88517 [==============================] - 393s 4ms/sample - loss: 3.2206 - val_loss: 3.3955
Epoch 22/50
88517/88517 [==============================] - 392s 4ms/sample - loss: 3.1973 - val_loss: 3.3844
```

3.5 Prediction

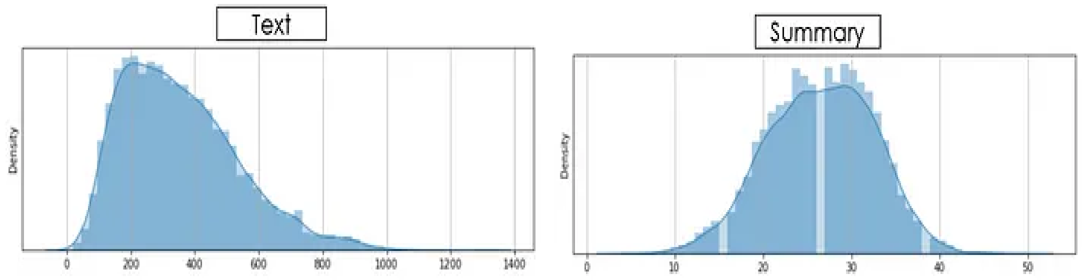
Predicting in machine learning is the act of using a trained model to predict or estimate data that has not yet been seen or that will be observed in the future based on patterns and correlations discovered from the training data. It is a fundamental aspect of many machine learning tasks, such as classification, regression, and even text summarization.When a model is trained, it learns the underlying patterns and dependencies in the training data. In order to reduce the discrepancy between the anticipated output and the actual output of the training data, the model's internal parameters are adjusted as part of the learning process. Once the model is trained, it can be used to predict the outcomes for new, unseen data. The prediction process involves feeding the input data into the trained model and obtaining the predicted output. The input data can take various forms depending on the problem domain, such as numerical features, images, or textual data. For text summarization, the input data typically consists of textual documents or articles.The output of the model's prediction can take different forms depending on the specific text summarization approach. It could be a summary sentence, a sequence of sentences, or even a hierarchical representation of the original text.



```
Review: pope francis on tuesday called for respect for each ethnic group in speech delivered in myanmar avoiding reference to th
e rohingya minority community as the nation works to restore peace the healing of wounds must be priority he said the pope myanm
ar visit comes amid the country military crackdown resulting in the rohingya refugee crisis
Original summary: start pope avoids mention of rohingyas in key myanmar speech end
Predicted summary:  start pope urges myanmar to rohingya violence end


Review: students of government school in uttar pradesh sambhal were seen washing dishes at in school premises on being approache
d basic shiksha adhikari virendra pratap singh said yes have also received this complaint from elsewhere we are inquiring and ac
tion will be taken against those found guilty
Original summary: start students seen washing dishes at govt school in up end
Predicted summary:  start up school students caught on camera end


Review: apple india profit surged by 140 in 2017 18 to crore compared to ãçâ â'373 crore in the previous fiscal the indian unit
of the us based company posted 12 growth in revenue last fiscal at ãçâ â'13 crore apple share of the indian smartphone market dr
opped to 1 in the second quarter of 2018 according to counterpoint research
Original summary: start apple india profit rises 140 to nearly ãçâ â'900 crore in fy18 end
Predicted summary:  start apple india profit rises to crore in march quarter end


Review: uber has launched its electric scooter service in santa monica us at 1 to unlock and then 15 cents per minute to ride it
comes after uber acquired the bike sharing startup jump for reported amount of 200 million uber said it is branding the scooters
with jump for the sake of consistency for its other personal electric vehicle services
Original summary: start uber launches electric scooter service in us at 1 per ride end
Predicted summary:  start uber launches its first electric taxi service in us end


Review: around 80 people were injured in accidents related to kite flying during celebrations of makar sankranti in rajasthan ja
ipur officials said the victims included those who fell while flying kites and those injured by glass coated kite string officia
ls added meanwhile around 100 birds were reported to be injured by between january 13 and 15
Original summary: start 80 people injured in flying related accidents in jaipur end
Predicted summary:  start 2 killed injured in explosion at railway station in rajasthan end


Review: uk entrepreneur richard browning has announced the launch of his startup gravity which has created flight jet powered su
it that will be priced at about ãçâ â'1 3 crore the suit has custom built exoskeleton with six attached micro jet engines fuelle
d by kerosene from backpack browning claims the can travel at speed of up to 450 kmph
Original summary: start startup makes ãçâ â'1 3 crore jet powered flying suit end
Predicted summary:  start startup makes flying car that can be drunk in time end
```

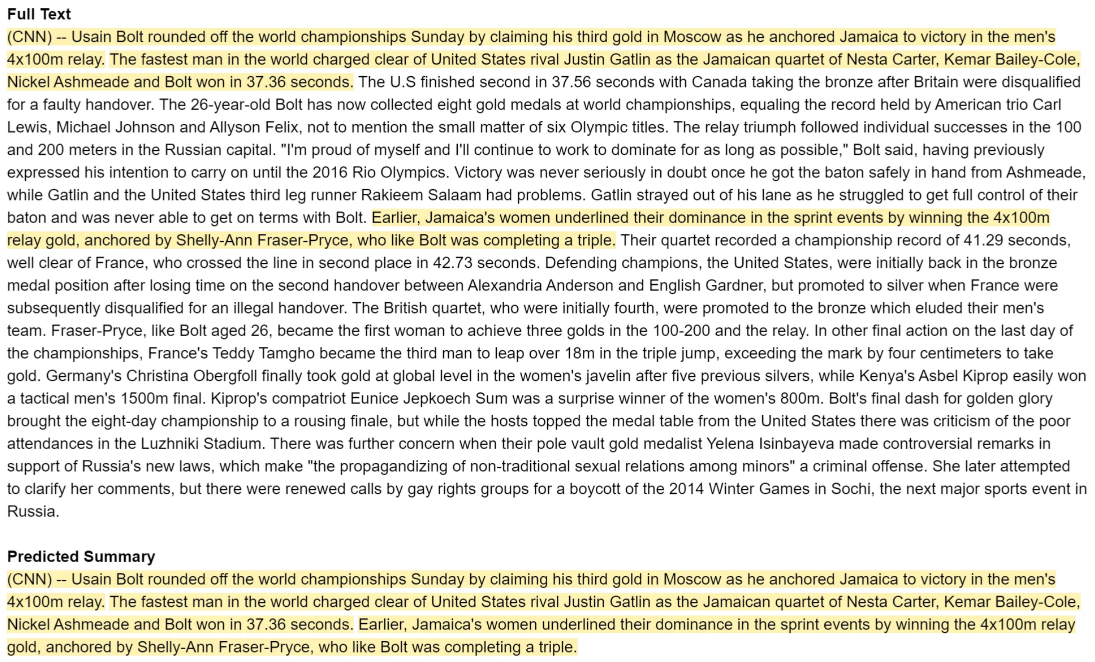
3.6 Results

In result, we are using rouge metrics for find the performance of the method. Based on this we test a semple text for finding the summary of sample.In the following figure, we make a graph for text and summary lengths from the data set for better descriptions of results.



here, input text has average length of 200 words and summary average length is about 35 words depending on relevant content on the input text.

For a simple test we input a CNN new article for finding the text summarization performance using ROUGE matrics.

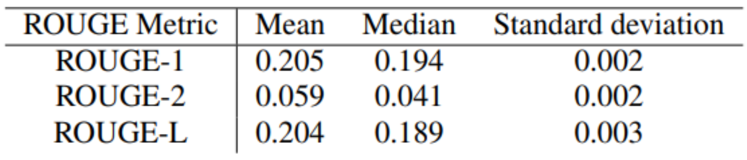


Full Text

(CNN) -- Usain Bolt rounded off the world championships Sunday by claiming his third gold in Moscow as he anchored Jamaica to victory in the men's 4x100m relay. The fastest man in the world charged clear of United States rival Justin Gatlin as the Jamaican quartet of Nesta Carter, Kemar Bailey-Cole, Nickel Ashmeade and Bolt won in 37.36 seconds. The U.S finished second in 37.56 seconds with Canada taking the bronze after Britain were disqualified for a faulty handover. The 26-year-old Bolt has now collected eight gold medals at world championships, equaling the record held by American trio Carl Lewis, Michael Johnson and Allyson Felix, not to mention the small matter of six Olympic titles. The relay triumph followed individual successes in the 100 and 200 meters in the Russian capital. "I'm proud of myself and I'll continue to work to dominate for as long as possible," Bolt said, having previously expressed his intention to carry on until the 2016 Rio Olympics. Victory was never seriously in doubt once he got the baton safely in hand from Ashmeade, while Gatlin and the United States third leg runner Rakieem Salaam had problems. Gatlin strayed out of his lane as he struggled to get full control of their baton and was never able to get on terms with Bolt. Earlier, Jamaica's women underlined their dominance in the sprint events by winning the 4x100m relay gold, anchored by Shelly-Ann Fraser-Pryce, who like Bolt was completing a triple. Their quartet recorded a championship record of 41.29 seconds, well clear of France, who crossed the line in second place in 42.73 seconds. Defending champions, the United States, were initially back in the bronze medal position after losing time on the second handover between Alexandria Anderson and English Gardner, but promoted to silver when France were subsequently disqualified for an illegal handover. The British quartet, who were initially fourth, were promoted to the bronze which eluded their men's team. Fraser-Pryce, like Bolt aged 26, became the first woman to achieve three golds in the 100-200 and the relay. In other final action on the last day of the championships, France's Teddy Tamgho became the third man to leap over 18m in the triple jump, exceeding the mark by four centimeters to take gold. Germany's Christina Obergfoll finally took gold at global level in the women's javelin after five previous silvers, while Kenya's Asbel Kiprop easily won a tactical men's 1500m final. Kiprop's compatriot Eunice Jepkoech Sum was a surprise winner of the women's 800m. Bolt's final dash for golden glory brought the eight-day championship to a rousing finale, but while the hosts topped the medal table from the United States there was criticism of the poor attendances in the Luzhniki Stadium. There was further concern when their pole vault gold medalist Yelena Isinbayeva made controversial remarks in support of Russia's new laws, which make "the propagandizing of non-traditional sexual relations among minors" a criminal offense. She later attempted to clarify her comments, but there were renewed calls by gay rights groups for a boycott of the 2014 Winter Games in Sochi, the next major sports event in Russia.

Predicted Summary

(CNN) -- Usain Bolt rounded off the world championships Sunday by claiming his third gold in Moscow as he anchored Jamaica to victory in the men's 4x100m relay. The fastest man in the world charged clear of United States rival Justin Gatlin as the Jamaican quartet of Nesta Carter, Kemar Bailey-Cole, Nickel Ashmeade and Bolt won in 37.36 seconds. Earlier, Jamaica's women underlined their dominance in the sprint events by winning the 4x100m relay gold, anchored by Shelly-Ann Fraser-Pryce, who like Bolt was completing a triple.



ROUGE Metric	Mean	Median	Standard deviation
ROUGE-1	0.205	0.194	0.002
ROUGE-2	0.059	0.041	0.002
ROUGE-L	0.204	0.189	0.003

REFERENCES

[1] Luhn, Hans Peter. "The automatic creation of literature abstracts." *IBM Journal of research and development* 2.2 (1958): 159-165.

[2] Mani, Inderjeet, and Mark T. Maybury, eds. *Advances in automatic text summarization*. MIT press, 1999.

[3] McKeown, K., & Radev, D. R. (1995, July). Generating summaries of multiple news articles. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 74-82).

[4] Erkan, Günes, and Dragomir R. Radev. "Lexrank: Graph-based lexical centrality as salience in text summarization." *Journal of artificial intelligence research* 22 (2004): 457-479.

[5] Cheung, J. C. K., & Penn, G. (2013, August). Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1233-1242).

[6] Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., & Huang, X. (2020). Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.

[7] Gupta, S., & Gupta, S. K. (2019). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, *121*, 49-65.

[8] Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, *1*, 43-52.

[9] Wallach, H. M. (2006, June). Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning* (pp. 977-984).

[10] Tsai, C. F. (2012). Bag-of-words representation in image annotation: A review. *International Scholarly Research Notices*, *2012*.

[11] Liu, Y., Liu, Z., Chua, T. S., & Sun, M. (2015, February). Topical word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 29, No. 1).

[12] Li, Q., Shah, S., Liu, X., & Nourbakhsh, A. (2017, May). Data sets: Word embeddings learned from tweets and general data. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1, pp. 428-436).

[13] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, *13*(4), 18-28.

[14] Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. In *Machine learning* (pp. 101-121). Academic Press.

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

[16] Wołk, K., & Marasek, K. (2015). Enhanced bilingual evaluation understudy. *arXiv preprint arXiv:1509.09088*.

[17] Khan, S. N., & Usman, I. (2019). Amodel for english to urdu and hindi machine translation system using translation rules and artificial neural network. *Int. Arab J. Inf. Technol.*, *16*(1), 125-131.

[18] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, Chennai, India, 2017, pp. 1-6, doi: 10.1109/ICCCSP.2017.7944061.

[19] Juan-Manuel TORRES-MORENO. Automatic Text Summarization. London: ISTE, 2014. isbn: 978-1-84821-668-6.

[20] Juan-Manuel TORRES-MORENO. Automatic Text Summarization. London: ISTE, 2014. isbn: 978-1-84821-668-6. [2] Petr MACHOVEC. "Automatická sumarizace textu [online]". Master's thesis. Masaryk University, Faculty of Informatics, Brno, 2015 [cit. 2016-05-10]. url: http://is.muni.cz/th/359331/fi_ m/. [3] Ani NENKOVA. "Automatic Summarization". In: Foundations and Trends in Information Retrieval 5 (2 2011). [4] Charu C. AGGARWAL and ChengXiang ZHAI. Mining Text Data. New York: Springer, 2012, pp. 43–76. isbn: 978-1-4614-3222-7.

[21] Neto, J. L., Freitas, A. A., & Kaestner, C. A. (2002). Automatic text summarization using a machine learning approach. In *Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002 Porto de Galinhas/Recife, Brazil, November 11–14, 2002 Proceedings 16* (pp. 205-215). Springer Berlin Heidelberg.

[22] Miles Osborne. 2002. Using maximum entropy for sentence extraction. In Proceedings of the ACL-02 Workshop on Automatic Summarization, pages 1–8, Phildadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[23] Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404-411).

[24] Tsai, C. I. (2019). A Study on Neural Network Modeling Techniques for Automatic Document Summarization.

[25] Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

[26] Abdel-Salam, S., & Rafea, A. (2022). Performance study on extractive text summarization using BERT models. *Information*, *13*(2), 67.

[27] Srikanth, A., Umasankar, A. S., Thanu, S., & Nirmala, S. J. (2020, October). Extractive text summarization using dynamic clustering and co-reference on BERT. In *2020 5th International Conference on Computing, Communication and Security (ICCCS)* (pp. 1-5). IEEE.

[28] Gupta, S., & Gupta, S. K. (2019). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, *121*, 49-65.

[29] Lin, H., & Ng, V. (2019, July). Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 9815-9822).

[30] Liu, F., Flanigan, J., Thomson, S., Sadeh, N., & Smith, N. A. (2018). Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399*.

[31] Nallapati, R., Xiang, B., & Zhou, B. (2016). Sequence-to-sequence rnns for text summarization.

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

[33] Nallapati, R., Xiang, B., & Zhou, B. (2016). Sequence-to-sequence rnns for text summarization.

[34] Yao, K., Zhang, L., Du, D., Luo, T., Tao, L., & Wu, Y. (2018). Dual encoding for abstractive text summarization. *IEEE transactions on cybernetics*, *50*(3), 985-996.

[35] Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *15*(4), 531-538.