# EXTRACTIVE TEXT SUMMARIZATION

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY (M. TECH)

**IN**

**ARTIFICIAL INTELLIGENCE (AI)**

Submitted by

**ABHISHEK KUMAR (2K21/AFI/23)**

Under the Supervision of

**MS. GARIMA CHHIKARA**

**ASSISTANT PROFESSOR**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**DELHI TECHNOLOGICAL UNIVERSITY**
**(Formerly Delhi College of Engineering)**
**Bawana Road, Delhi 110042**
**MAY, 2023**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**DELHI TECHNOLOGICAL UNIVERSITY**
**(Formerly Delhi College of Engineering)**
**Bawana Road, Delhi-110042**

# CANDIDATE'S DECLARATION

I, **Abhishek Kumar, 2K21/AFI/23** students of M.Tech, hereby declare that the project Dissertation titled "**Extractive Text Summarization**" which is submitted by us to the **Department of Computer Science and Engineering, Delhi Technological University, Delhi** in partial fulfilment of the requirement for the award of 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** **Abhishek Kumar**
Date: **31th May,2023** **2K21/AFI/23**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**DELHI TECHNOLOGICAL UNIVERSITY**
**(Formerly Delhi College of Engineering)**
**Bawana Road, Delhi-110042**

# CERTIFICATE

I hereby certify that the Project Dissertation titled "**Extractive Text Summarization**" which is submitted by **Abhishek Kumar, 2K21/AFI/23**, **Department of Computer Science and Engineering**, **Delhi Technological University, Delhi** in partial fulfilment of the requirement for the award of the 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: **31th May,2023**

**Ms. Garima Chhikara**
**Assistant Professor**

# ABSTRACT

Over the past decade, there has been remarkable growth in the domains of Artificial Intelligence (AI), Machine Learning (ML), and Data Science, presenting vast opportunities in diverse industries such as healthcare, finance, and transportation. Notably, the field of Natural Language Processing (NLP), a branch of AI and ML, has experienced significant advancements. NLP involves the machine-based processing and understanding of human language. Among its various applications, text summarization holds prominence as it enables machines to condense lengthy texts into concise summaries. This project highlights the utilization of multiple extractive text summarization techniques, including BERT, GPT-2, KL-summarizer, Luhn, LEX, and Word Rank. The resultant extractive summaries are then evaluated against human-generated summaries using three distinct scoring methods: Rouge Score, BERT Score, and Mover Score. Through this project, we demonstrate the efficacy of these techniques in generating summaries and assess their quality by comparing them against summaries produced by humans using the specified scoring metrics.

*Keywords- Artificial Intelligence, Natural Language Processing, BERT (Bi-directional Encoder Representation from Transformers), GPT-2(Generative Pretrained Transformer), KL-summarizer, Luhn, LEX and Word Rank, ROUGE Score, BERT Score and MoverScore.*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**DELHI TECHNOLOGICAL UNIVERSITY**
**(Formerly Delhi College of Engineering)**
**Bawana Road, Delhi-110042**

# ACKNOWLEDGEMENT

I wish to express our sincerest gratitude to **Ms. Garima Chhikara** for his continuous guidance and mentorship that he provided us during the project. She showed us the path to achieve our targets by explaining all the tasks to be done and explained to us the importance of this project as well as its industrial relevance. She was always ready to help me and clear my doubts regarding any hurdles in this project. Without her constant support and motivation, this project would not have been successful.

**Abhishek Kumar**
**(2K21/AFI/23)**

# TABLE OF CONTENT

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS AND EQUATIONS

## ABBREVIATION

1. NLP - NATURAL LANGUAGE PROCESSING
2. TF-IDF- TERM FREQUENCY INVERSE DOCUMENT FREQUENCY
3. RNN - RECURRENT NEURAL NETWORK
4. LSTM - LONG SHORT-TERM MEMORY
5. BERT- BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS
6. MLM - MASKED LANGUAGE MODEL
7. NSP - NEXT SENTENCE PREDICTION
8. GPU - GRAPHICS PROCESSING UNIT
9. TPU - TENSOR PROCESSING UNIT
10. GPT-2 – GENERATIVE PRETRAINED TRANSFORMER
11. KL - KULLBACK-LEIBLER DIVERGENCE
12. ROUGE - RECALL-ORIENTED UNDERSTUDY FOR GISTING EVALUATION
13. WMD - WORD MOVER'S DISTANCE

## EQUATIONS

# CHAPTER 1

# INTRODUCTION

Text summary is the process of reducing the amount of information in a text, such as an article, paper, or web page, while keeping the essential points and important details. Text summarization seeks to convey a clear, cogent summary of the original text that conveys its main ideas.

Text summarization can be classified into two main types: extractive text summarization and abstractive text summarization. These approaches differ in how they generate summaries from given texts.

Extractive Summarization: The most significant sentences or phrases from the source material are chosen and extracted using this technique to create a summary. The algorithms used in extractive summarization techniques frequently rate phrases according to their applicability, significance, or salience. To generate the summary, the chosen sentences are then put together.

Abstractive Summarization: Abstractive summarization seeks to provide a summary by comprehending the original text and conveying the key concepts in a clear, succinct manner, even if the summary includes information that isn't explicitly stated in the original text. This strategy uses deep learning models and natural language generation to produce summaries of the provided text that are more like human speech.

Here are some differences between Extractive Text summarization and Abstractive Text summarization.

| **Extractive Text summarization** | **Abstractive Text summarization** |
|---|---|
| In order to provide a summary, extractive summarization chooses and rearranges already sentences or phrases from the original material. It uses sentence scoring algorithms to evaluate each sentence's significance before choosing the top-scoring sentences for the summary. | In order to effectively summarise an original text, abstractive summarization seeks to create new phrases that do so. It entails comprehending the content and context of the material and coming up with creative language to communicate the essential ideas. Techniques for generating natural language naturally used in this process include neural networks or language models. |

| | |
|---|---|
| Sentences that have been taken verbatim from the source text make up the extractive summarization's output. A portion of the original sentences have been rearranged or changed for coherence to create the summary. | A summary produced using abstractive summarization may include sentences that were absent from the original text. The programme creates new phrases that effectively convey the main ideas while rephrasing or rearranging the material. |
| Extractive methods tend to preserve the original information present in the source text. The summary is a concise representation of the most salient sentences, but it may not capture the entire context or provide additional insights. | Abstractive techniques have the ability to provide summaries that go beyond the original text and effectively convey the topic. Even though they weren't specifically mentioned in the original text, they can introduce new phrases that communicate the important information. |
| Extractive methods face challenges in maintaining coherence and flow in the summary since sentences are selected independently. They might struggle to handle information that requires combining multiple sentences or paraphrasing. | Abstractive methods face challenges in generating linguistically and contextually accurate sentences. They need to overcome difficulties related to language understanding, semantic representation, and ensuring the generated sentences are coherent and faithful to the original text. |
| Extractive methods are generally less complex computationally since they rely on sentence scoring and selection rather than generating new sentences. They can be more computationally efficient for large-scale summarization tasks. | Abstractive methods involve more complex natural language processing and generation techniques, often utilizing neural networks or language models. They require more computational resources and can be slower compared to extractive methods. |

Both extractive and abstractive summarization have their own strengths and weaknesses. While abstractive summarising has the ability to provide more human-like summaries and offer a better comprehension of the topic, extractive summary is simpler and maintains the original text. The task's precise needs and the trade-offs between accuracy, fluency, and efficiency will determine which option is best.

## 1.1. OBJECTIVE

• Outline a few extractive text summarization algorithms that can pick out the most pertinent phrases from a given textual unit.

• To generate summaries, our strategy involves utilizing several techniques and models, namely Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2, and BERT. These approaches and models have been selected with the aim of achieving effective summarization outcomes.

• Furthermore, we showcase the quantitative value of our proposed methodology by employing various metrics, namely ROUGE, BERT, and MoverScore. These metrics serve as objective measures to evaluate the effectiveness and performance of our approach. By utilizing these quantitative assessment tools, we can quantitatively assess the quality and efficacy of our suggested method in generating summaries.

## 1.2. APPLICATION

Extractive text summarising is a method for creating a brief summary of a lengthy text by highlighting the key clauses or phrases in the original text. It has several practical applications across different domains. Here are some common applications of extractive text summarization:

1. News Summarization: By automatically creating summaries of news items, extractive summarization enables readers to rapidly understand a story's essential points without having to read the full piece. News aggregation platforms often use extractive summarization to provide users with a brief overview of various news stories.

2. Document Summarization: Extractive summarization can help summarize lengthy documents, reports, or research papers. It enables users to get a quick overview of the document's content and key findings. Researchers, students, or other professions who need to process a lot of information may find this to be extremely helpful.

3. Social Media Summarization: Extractive summarization techniques can be employed to summarize social media posts or threads, such as Twitter feeds or online discussions. By extracting the most relevant and informative sentences, it becomes easier to comprehend the overall sentiment, trending topics, or important updates from a large volume of social media content.

4. Legal Document Analysis: Extractive text summarization can assist legal professionals in analysing and summarizing legal documents, such as court rulings, contracts, or legal opinions. By extracting key sentences or sections, lawyers can quickly identify important arguments, rulings, or clauses without having to read the entire document.

5. Customer Feedback Analysis: Extractive summarization can be applied to analyse customer feedback, reviews, or survey responses. By extracting important sentences or phrases, businesses can gain insights into customer sentiment, identify recurring issues, or extract actionable information for product improvements or marketing strategies.

6. Meeting Summaries: Extractive summarization can aid in summarizing the minutes or transcripts of business meetings, conferences, or interviews. It helps in capturing the main discussion points, decisions, and action items, saving time for participants who may need to review or reference the meeting outcomes later.

7. E-commerce Product Descriptions: In e-commerce platforms, extractive summarising techniques may be utilised to provide succinct summaries of product descriptions. By extracting the most relevant features, benefits, and customer reviews, it helps potential buyers quickly understand the key attributes of a product before making a purchase decision.

These are just a few examples of how extractive text summarization can be applied across various domains to efficiently process and comprehend large volumes of textual data.

## 1.3. USAGE

Extractive text summarization has various applications and can be used in different scenarios to help extract important information from a given text. Here are some specific use cases and examples of how extractive text summarization can be utilized:

1. Content Aggregation: Extractive summarization can be employed by news aggregators or content curation platforms to create short summaries of articles from multiple sources. This

allows users to quickly browse through summaries and decide which articles they want to read in detail.

2. Search Engine Result Summaries: Search engines can use extractive summarization to generate snippets or summaries for search results. These summaries provide users with a brief overview of the content on a webpage, helping them assess the relevance of the search result without clicking through to the full page.

3. Document Summarization: Extractive text summarization can be used to summarize lengthy documents, research papers, or reports. Researchers, students, or professionals can save time by quickly reviewing the summarized content and identifying the key points without having to read the entire document.

4. Social Media Monitoring: Extractive summarization techniques can be applied to social media monitoring tools to summarize and analyse trending topics, discussions, or user-generated content. By extracting important sentences or phrases, companies can gain insights into public sentiment, identify emerging trends, or monitor the impact of their brand or products on social media.

5. Email Summarization: Extractive text summarization can help individuals manage their email overload by automatically summarizing the content of incoming emails. This allows users to quickly prioritize and respond to important messages without spending excessive time on each email.

6. Legal Case Analysis: Extractive summarization can assist legal professionals in analyzing and summarizing legal cases, court rulings, or lengthy legal documents. By extracting key arguments, legal precedents, or important sections, lawyers can efficiently review and understand the essence of complex legal texts.

7. Audio and Video Transcription: Extractive summarization techniques can be utilized to summarize audio or video recordings, such as lectures, interviews, or conference presentations. By extracting important sentences or segments, users can quickly review the content and find relevant information without listening to the entire recording.

8. Chatbot Responses: Chatbots and virtual assistants can use extractive summarization to generate concise and informative responses to user queries. By extracting key information from a knowledge base or a database of frequently asked questions, chatbots can provide quick and relevant answers to users.

These are just a few examples highlighting the practical applications of extractive text summarization. The method may be used in a wide range of situations and sectors where it is advantageous to extract key details or provide succinct summaries.

# CHAPTER 2

# LITERATURE REVIEW

A natural language processing (NLP) approach called extractive text summarising seeks to provide a brief summary of a given document by extracting the key words or sentences. Contrasting with abstractive summarising, which generates summaries by paraphrasing and rephrasing the information, is this method. The history of extractive summarization is lengthy, and it has made tremendous strides over time. I'll give a thorough account of the development of extractive text summarization in this response.

Early Approaches: The origins of extractive text summarization may be found in the first NLP studies. In the 1950s and 1960s, researchers explored rule-based methods to automatically summarize texts. These approaches involved using linguistic rules and heuristics to identify key sentences based on sentence length, word frequency, and position within the document. However, these early attempts faced challenges in dealing with the nuances of language and producing coherent summaries.

Statistical Approaches: In extractive summarization, statistical approaches became more popular in the 1990s. To evaluate phrases according to their relevance, researchers started using algorithms like Term Frequency-Inverse Document Frequency (TF-IDF) and cosine similarity. In order to identify important phrases, TF-IDF applies weights to words based on their frequency inside a document and throughout the corpus. Based on a vector representation of the TF-IDF weights in the sentences, cosine similarity calculates how similar the phrases are. By employing these techniques, researchers were able to extract sentences that were most similar to the overall content of the document.

Graph-Based Approaches: Graph-based techniques became an important new area of study for extractive text summarization. In these methods, sentences are visualised as nodes in a graph, with the similarity between phrases represented by the edges. Sentences were ranked based on their centrality and relevance inside the network using algorithms like PageRank, which were influenced by Google's web page ranking algorithm. By applying graph-based algorithms,

researchers were able to identify key sentences that were well-connected to other important sentences, thus providing a more coherent summary.

Supervised Machine Learning: With the advent of machine learning techniques, supervised methods for extractive text summarization gained traction. Researchers started using labelled datasets, where human-generated summaries were paired with their corresponding source documents. An important new area of research for extractive text summarization is graph-based approaches. In these techniques, sentences are represented as nodes in a graph, with edges signifying phrase similarity. Using methods like PageRank, which were influenced by Google's web page ranking algorithm, sentences were rated according to their centrality and significance within the network. These models were trained on large corpora and demonstrated improved performance over previous techniques.

Deep Learning and Neural Networks: The discipline of NLP, especially extractive summarization, underwent a revolution in the 2010s with the emergence of deep learning. To successfully mimic the sequential character of phrases, recurrent neural networks (RNNs), especially the Long Short-Term Memory (LSTM) form, were used. By processing the source document sequentially and encoding the information in a hidden state, LSTM networks were able to capture contextual dependencies and learn sentence representations. Extractive summarization's effectiveness was further enhanced by attention methods, including the Transformer model, which allowed the model to concentrate on pertinent sections of the material. These deep learning-based methods produced cutting-edge outcomes and showed their capacity to provide well-organized, useful summaries.

Transformer-Based Models: The introduction of the Transformer model in 2017 marked a significant milestone in NLP, and extractive summarization also benefited from this breakthrough. BERT (Bidirectional Encoder Representations from Transformers), one of the transformer models, excelled at extractive summarization among other NLP tasks. BERT-based models learn contextualised word representations by pretraining on sizable corpora, which are subsequently refined on datasets tailored for summarization [27].

# CHAPTER 3

# PROPOSED METHODOLOGY

## 3.1. BERT

One of the many natural languages processing (NLP) tasks that BERT (Bidirectional Encoder Representations from Transformers) has been successfully used to is text summarization. BERT is a potent language model. While BERT is primarily designed for contextual word representation and understanding, it can be adapted for extractive text summarization. In this article, we will explore BERT-based extractive text summarization in detail, discussing its steps, advantages, limitations, and potential future directions.

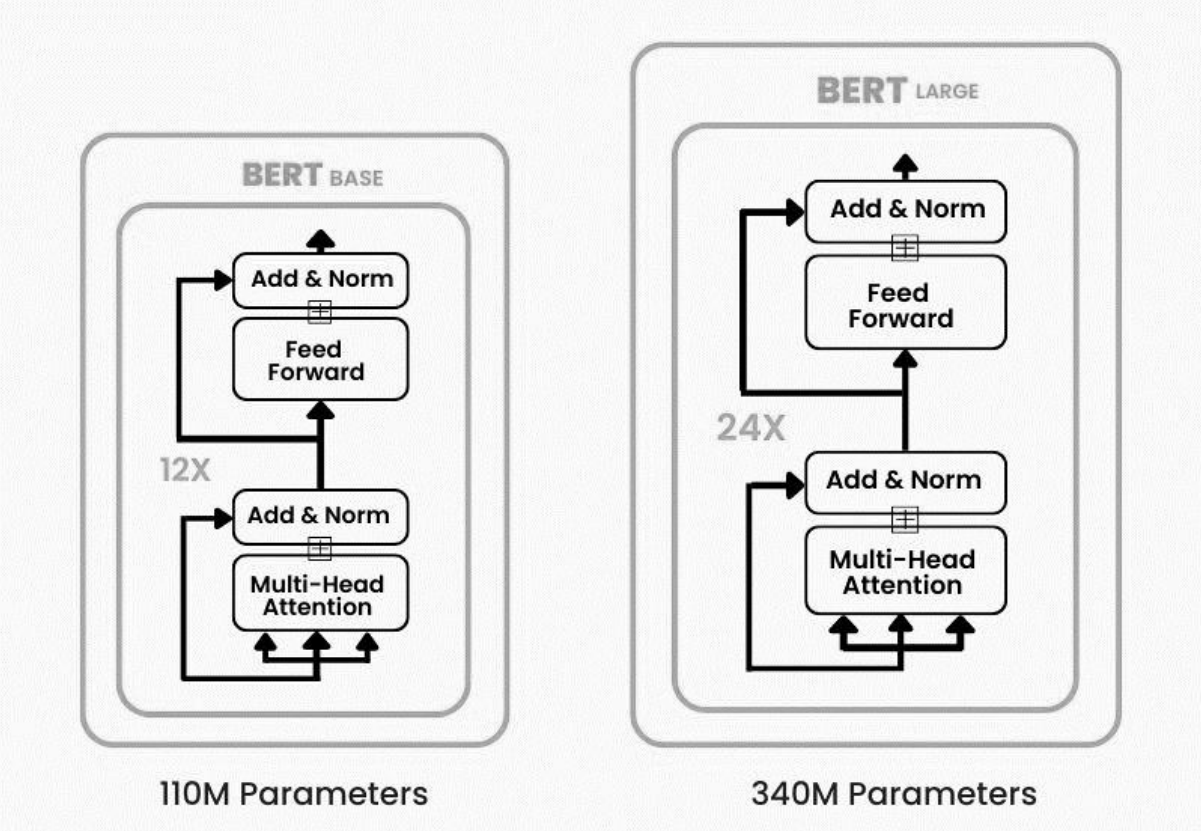



Fig 1- [1]

The pretrained language model BERT, released by Google Research in 2018, has transformed NLP tasks. It is built using the Transformer architecture, which enables it to recognise links between words and contextual information in a document. BERT models may be fine-tuned

for certain downstream tasks, such as text categorization, question-answering, and summarization, and are pretrained on substantial volumes of unlabelled text data.

In BERT-based extractive text summarising, key sentences or phrases from a document are selected in order to provide a summary.

### 3.1.1. WORKING OF BERT ALGORITHM

An outline of the key phases in the BERT-based extractive summarization procedure is provided below:

1. BERT was designed with a focus on handling larger amounts of text. The availability of vast and informative databases has greatly contributed to BERT's ability to comprehend various languages, including English. Training BERT using a larger dataset requires more time. The transformer architecture plays a crucial role in making the training of BERT feasible, and the utilization of Tensor Processing Units can accelerate the training procedure.

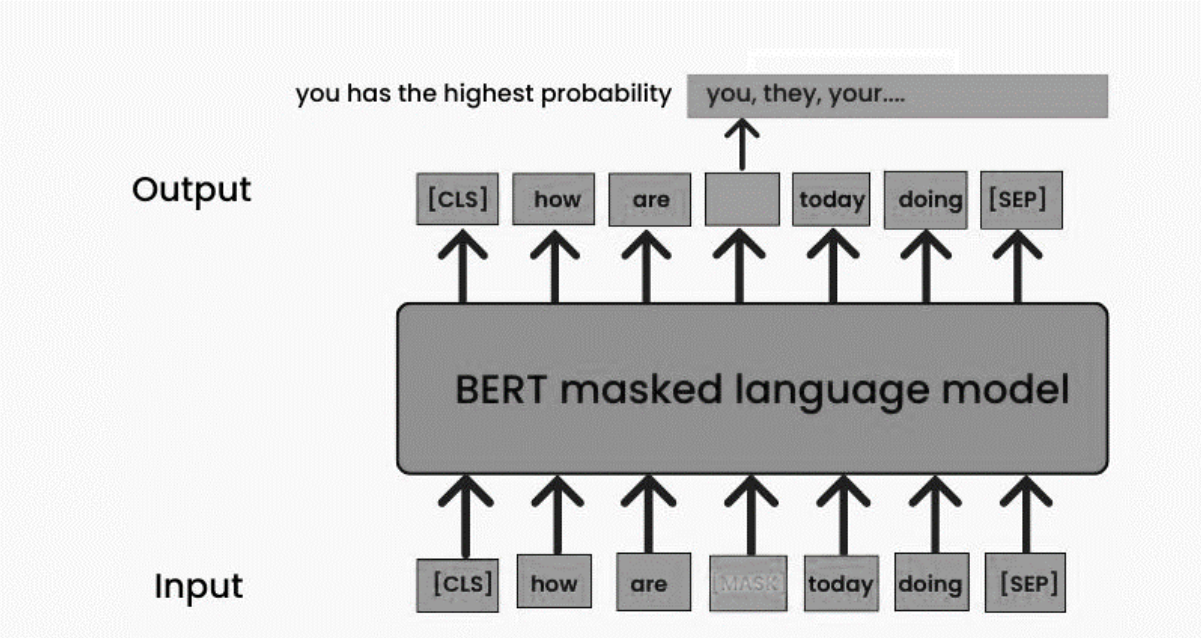



Fig 2 – [1]

2. Masked Language Model: By considering the surrounding words both preceding and following the concealed text, which provide contextual clues, we can make educated predictions about the absent word. The bidirectional method employed in this approach

greatly enhances the accuracy level. In the training process, approximately 15% of the tokenized words are randomly concealed, and BERT's objective is to infer the missing word.

3. Next Sentence Prediction: Next Sentence Prediction (NSP) is a method employed by BERT to comprehend the links between sentences by foretelling if a certain sentence will be followed by another. Half of the accurate predictions are rewarded with random utterances during training to increase BERT's accuracy, which helps BERT function better.

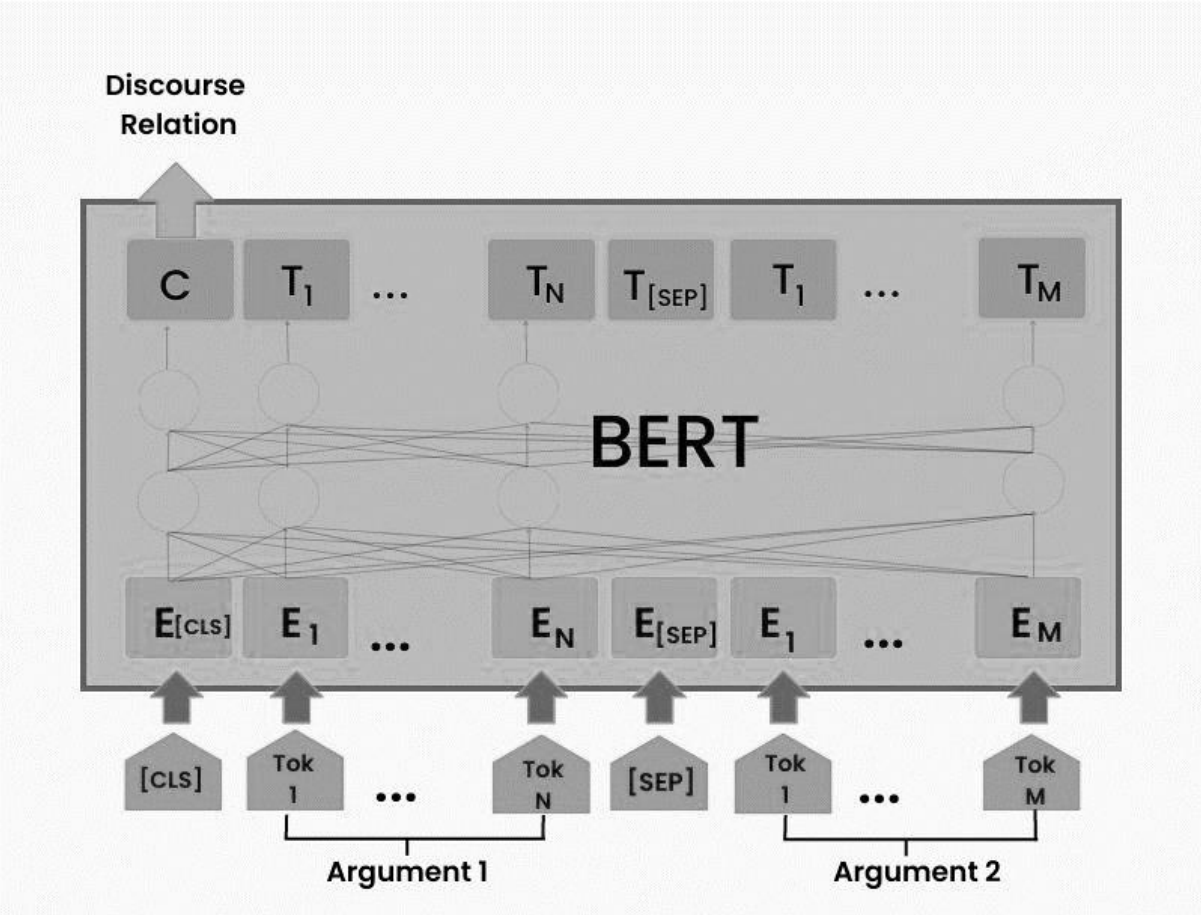



Fig 3 – [1]

4. Transformers: Machine learning training is effectively parallelized using the transformer architecture. The model can be trained fast on a large amount of data when we use massive parallelization. Transformers use attention to their advantage. It is an effective deep-learning technique that was originally used in computer vision models. Machine learning models must have the ability to focus on the most important information because human brains have limited memory. When the machine learning model achieves that, we may

prevent the wastage of computing resources and utilise them for processing irrelevant data. By providing signals to the words in a phrase that are important for subsequent processing, transformers establish differential weights [9].

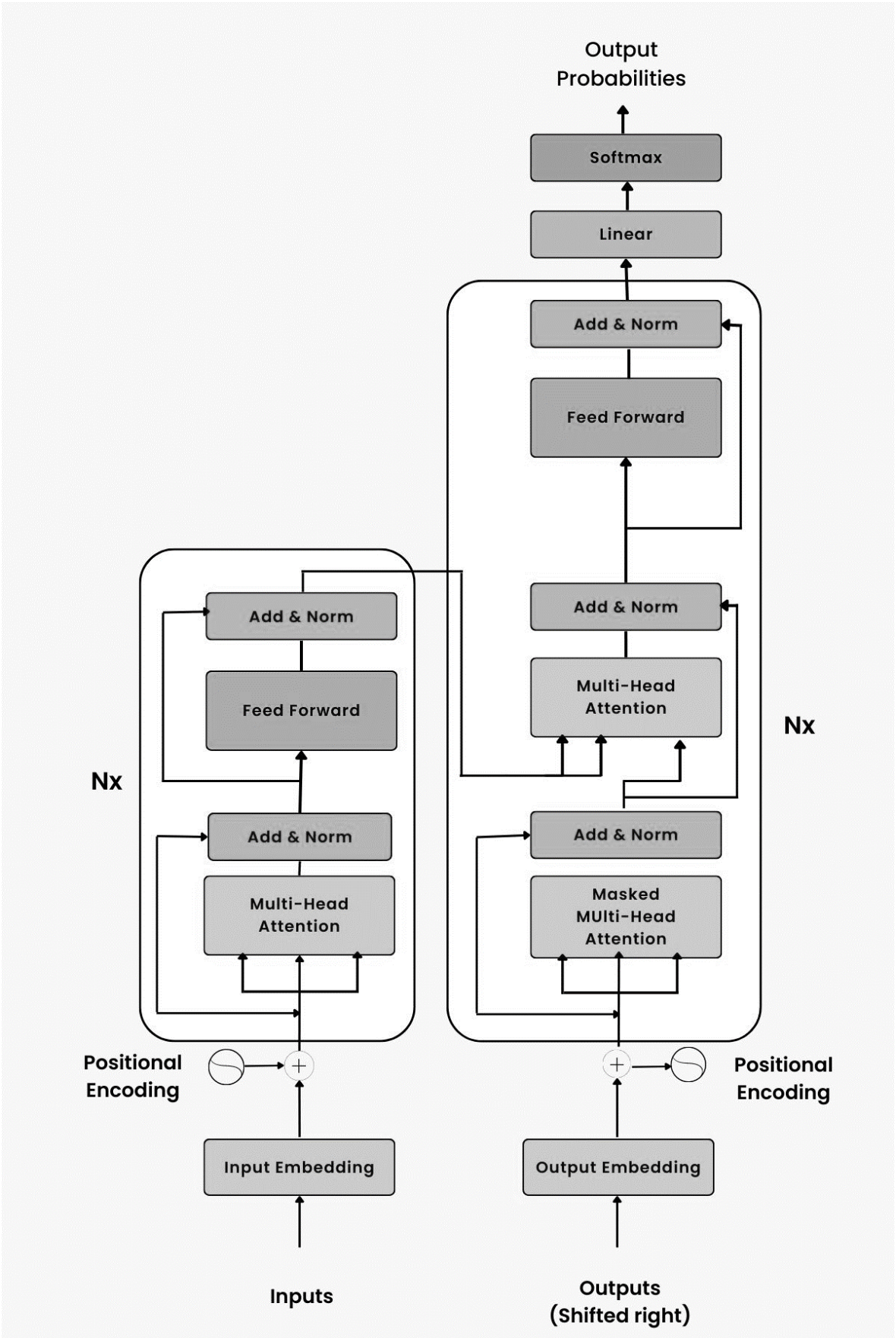

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked MUlti-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (Shifted right)

Fig 4 – [1]

In order to do this, a transformer must effectively process an input through levels of its transformer stack known as encoders. Decoders, a further stack of transformer layers, will aid in predicting the result. Transformers excel in unsupervised learning because they can handle more data points quickly.

5. Fine-tuning BERT

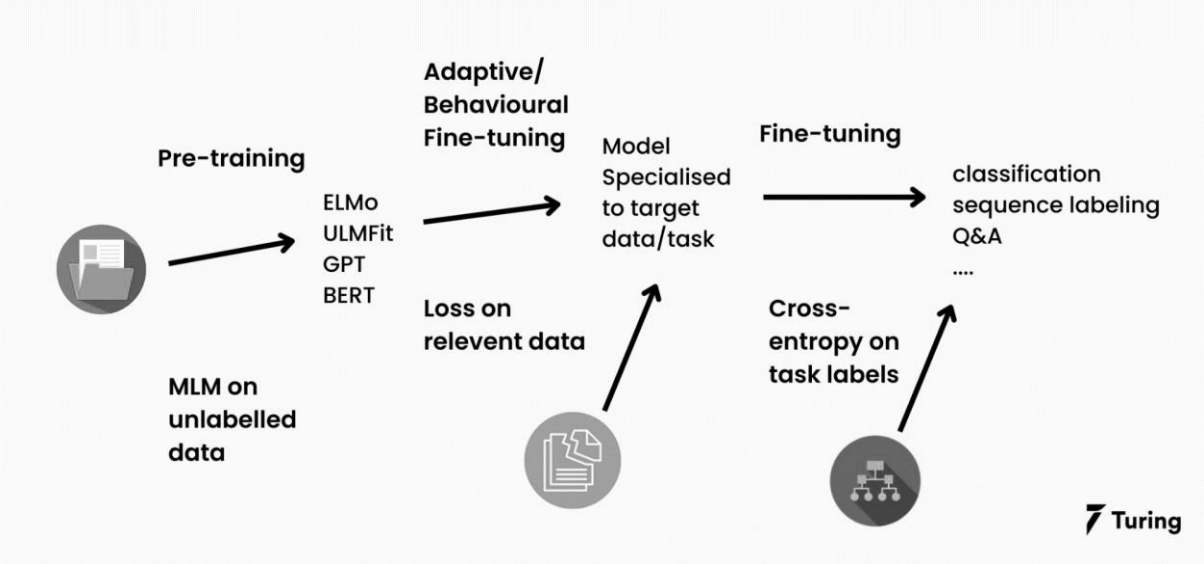



Fig 5 – [1]

It is usual practise to employ BERT (Bidirectional Encoder Representations from Transformers) to modify the pre-trained BERT model for certain downstream natural language processing (NLP) applications. BERT may use its overall language comprehension skills while acquiring knowledge tailored to a given job through fine-tuning.
Here's an overview of how fine-tuning BERT works:

1. Pre-training: Initial pre-training for BERT involves a sizable corpus of unlabelled text. BERT gains the ability to identify missing words in sentences (Masked Language Model) and comprehend the connections between phrases (Next Sentence Prediction) during pre-training. This pre-training phase aids BERT's acquisition of generic linguistic representations.

2. Task-specific data: To fine-tune BERT for a specific NLP task, you need a labelled dataset specific to that task. This dataset should consist of input texts and their corresponding labels or annotations.

3. Tokenization: The input texts in your task-specific dataset need to be tokenized into subword units called "word pieces." BERT uses the Word Piece tokenization scheme to break down words into smaller subword units. Tokenization ensures that the input texts align with BERT's vocabulary.

4. Model architecture: Each Transformer layer in the BERT system has a different set of attention heads. Contextual information is captured by the layers, and word dependencies are discovered by the attention heads. The model architecture stays the same as the pre-trained BERT during fine-tuning, but the weights are changed.

5. Fine-tuning procedure: Using task-specific data, the pre-trained BERT model's weights are updated during fine-tuning. The procedure typically consists of the following steps:

    - Input representation: The tokenized input texts are converted into numerical representations that BERT can process. This involves adding special tokens, segment IDs, and positional embeddings.

    - Forward pass: The input representations are fed through the BERT model, and the output representations are obtained. These representations capture contextualized information about the input texts.

    - Task-specific layers: Depending on the downstream task, additional task-specific layers can be added on top of BERT. These layers can include dense layers, convolutional layers, or recurrent layers, depending on the nature of the task.

    - Loss computation: The output from the task-specific layers is compared to the ground truth labels using a task-specific loss function (e.g., cross-entropy loss).

- Backpropagation and parameter updates: Gradient descent optimisation techniques like Adam or SGD are used to update the weights of all the layers, including BERT, as the loss propagates through the network.

- Iterative fine-tuning: Fine-tuning is an iterative process. The performance of the refined BERT model is assessed on a validation set, and the hyperparameters—such as learning rate, batch size, and number of training epochs—can be changed in accordance with the results. This process is repeated until satisfactory performance is achieved [8].

6. Inference: The BERT model may be adjusted and then used to draw conclusions from brand-new, untested data. The input texts are tokenized, converted into numerical representations, and passed through the model to obtain predictions or output representations, depending on the specific task.

It is vital to highlight that because to the enormous number of parameters in the model, fine-tuning BERT necessitates extensive computing resources, particularly potent GPUs or TPUs. the large number of parameters in the model. Additionally, fine-tuning BERT typically requires a substantial amount of labelled data specific to the target task for effective performance [7].

### 3.1.2. ADVANTAGES OF BERT

BERT-based extractive text summarization offers several advantages:

- Contextual Understanding: BERT's ability to capture contextual information and relationships between words helps in understanding the meaning and importance of sentences within a document. This leads to more accurate identification of important sentences for the summary.

- Semantic Representation: BERT's pretrained representations encode rich semantic information, allowing for a nuanced understanding of the text. This enables the model to capture intricate relationships between words and generate more coherent summaries.

- Generalization: BERT is pretrained on a vast amount of data from diverse sources, which helps it generalize well to different domains and styles of text. This makes BERT-based extractive summarization applicable to a wide range of documents.

- Fine-Tuning Flexibility: BERT models can be fine-tuned on specific summarization datasets or with additional task-specific objectives, allowing for better alignment with the summarization task at hand [1].

### 3.1.3. LIMITATIONS AND ENHANCEMENTS OF BERT

BERT-based extractive summarization also has certain limitations and challenges:

- Computational Resources: BERT models are computationally expensive, requiring significant computational resources and memory. Fine-tuning a BERT model for extractive summarization can be time-consuming and resource-intensive.

- Sentence Compression: Extractive summarization typically involves selecting sentences as they appear in the original document. However, these sentences may still contain redundant information. Additional techniques like sentence compression or paraphrasing may be required to further condense the summary.

- Lack of Abstractive Ability: BERT-based extractive summarization focuses on selecting and combining sentences, which limits its ability to generate novel or abstractive summaries. Abstractive summarization, which involves paraphrasing and generating new sentences, is a separate research direction.

Researchers are actively exploring enhancements and variations of BERT-based extractive text summarization. Some potential directions for improvement include:

- Joint Training: BERT-based models can be trained in a joint manner with other summarization-specific objectives, such as sentence compression, to improve the overall quality and coherence of the generated summaries.

- Transfer Learning: The effectiveness of extractive summarization can be improved by combining BERT models with other models or methodologies, such as graph-based approaches or reinforcement learning, to take use of their complementing qualities.

- Multimodal Summarization: Integrating BERT with multimodal information, such as images or videos associated with the text, can enable more comprehensive and informative summaries.

- Evaluation Metrics: Developing robust evaluation metrics specific to extractive summarization, beyond traditional metrics like ROUGE, can provide a better assessment of the quality, coherence, and informativeness of the generated summaries.

BERT-based extractive text summarization leverages the contextual understanding and semantic representation capabilities of BERT models to select important sentences from a document and generate a summary. Despite its limitations and challenges, BERT-based extractive summarization offers significant advantages in terms of contextual understanding, generalization, and flexibility.

As research in NLP progresses, ongoing advancements and improvements in BERT-based models, fine-tuning techniques, and evaluation metrics will continue to push the boundaries of extractive text summarization. With further developments, BERT-based approaches have the potential to play a crucial role in generating accurate and informative summaries from a wide range of textual data.

## 3.2. GPT-2

GPT-2 is primarily known for its capabilities in generating abstractive text rather than extractive summarization. While GPT-2 can be used for extractive summarization by ranking and selecting sentences based on their relevance, it is not specifically designed for this task. Extractive summarization techniques usually involve algorithms and models that focus on sentence selection based on saliency, coherence, and relevance.

A language model called GPT-2 (Generative Pre-trained Transformer 2) was created by OpenAI and launched in 2019. It is an expansion of its forerunner, GPT, that generates coherent

and contextually appropriate text using a transformer architecture with a considerable number of parameters (1.5 billion). GPT-2 was trained using a sizable corpus of freely accessible text obtained from the internet, enabling it to pick up on linguistic statistical trends and structures.

A language model is essentially a machine learning model that can analyse a portion of a phrase and predict the following word. This concept is covered in The Illustrated Word2vec [10]. Smartphone keyboards that suggest the next word based on what you've just written are the most well-known language models.

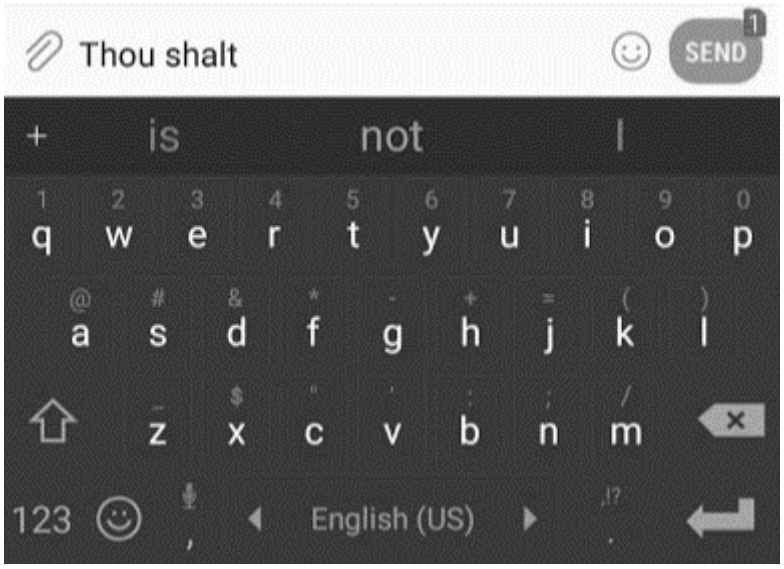



Fig 6- [2]

In this way, we can say that the GPT-2 is like a keyboard app's next word prediction tool, only much larger and more advanced than the keyboard app on your phone. The 40GB WebText dataset, which the OpenAI researchers scraped from the internet as part of their research, served as the basis for training the GPT-2. My preferred keyboard programme, SwiftKey, uses 78MBs of storage space, for comparison. The trained GPT-2's smallest form requires 500 MB of storage to keep all of its settings. Given that the biggest GPT-2 model is 13 times larger, it may use more than 6.5 GB of storage.

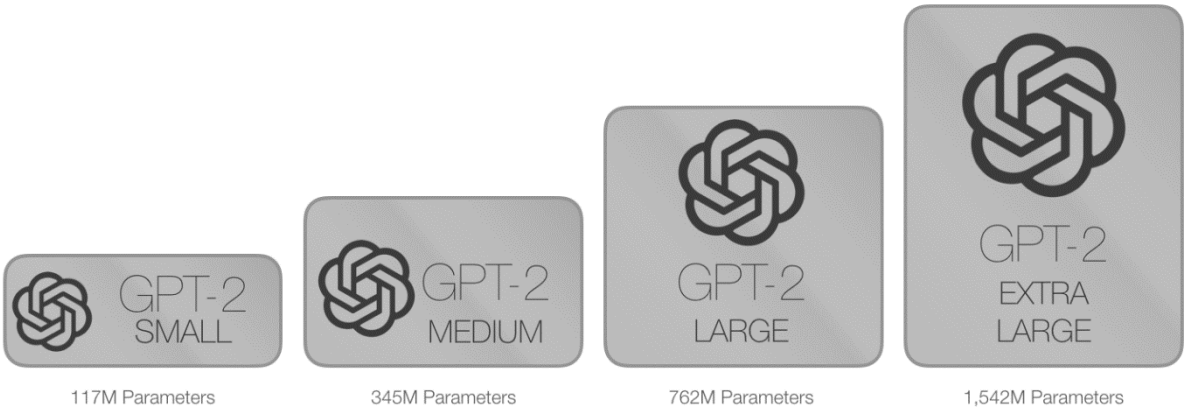



Fig 7 - [2]

The AllenAI GPT-2 Explorer [11] is a fantastic tool for GPT-2 experimentation. Ten potential predictions for the following word are shown (along with their likelihood score) using GPT-2. The next round of predictions will appear once you choose a word to continue composing the paragraph.

## 3.2.1. WORKING OF GPT- 2

### 3.2.1.1. LANGUAGE MODELING TRANSFORMER

The encoder and decoder, each of which is a stack of what we might refer to as transformer blocks, make up the original transformer model, as we have seen in The Illustrated Transformer [12]. Because the model dealt with machine translation, an issue where encoder-decoder designs have had success in the past, that architecture seemed acceptable.

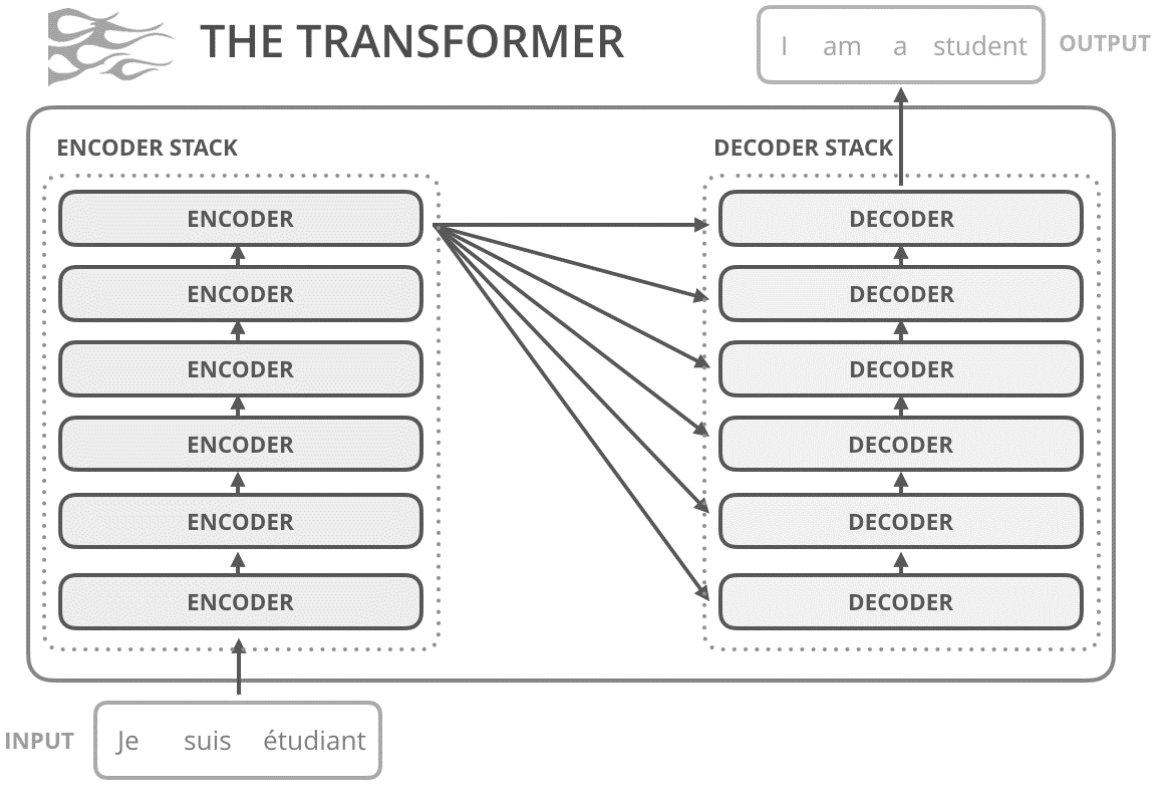



Fig 8 – [2]

Many of the succeeding studies' architectures employed simply one stack of transformer blocks instead of an encoder or a decoder. They were given massive volumes of training material, subjected to tremendous amounts of processing power, and stacked as high as was physically possible (some of these language models cost hundreds of thousands of dollars to train, and AlphaStar undoubtedly cost millions).

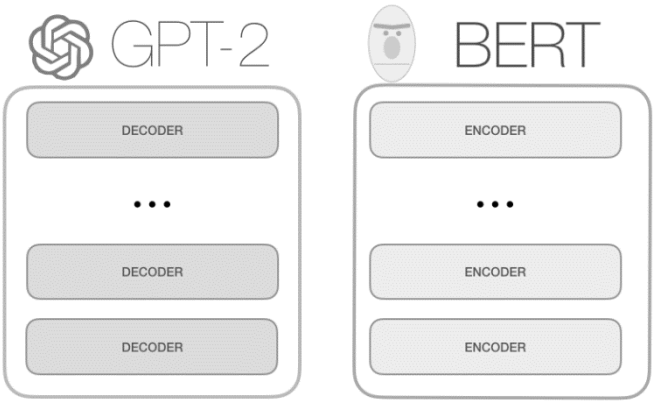



Fig 9 – [2]

It appears that one of the primary differences between the various GPT2 model sizes is this:

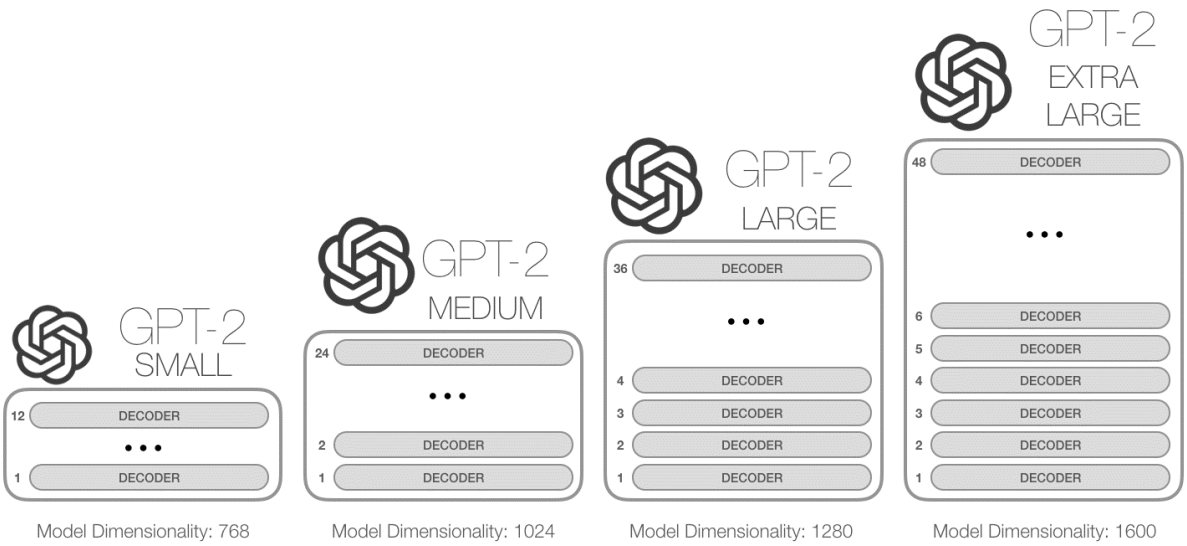



Fig 10 - [2]

### 3.2.1.2. GPT 2 AND BERT DIFFERENCE

Utilising transformer decoder blocks, the GPT-2 is constructed. Contrarily, BERT makes use of transformer encoder blocks. In the part that follows, we'll look at the distinction. However, one significant distinction between the two is that GPT2 outputs one token at a time, much like conventional language models. For instance, ask a skilled GPT-2 to repeat the first rule of robotics:

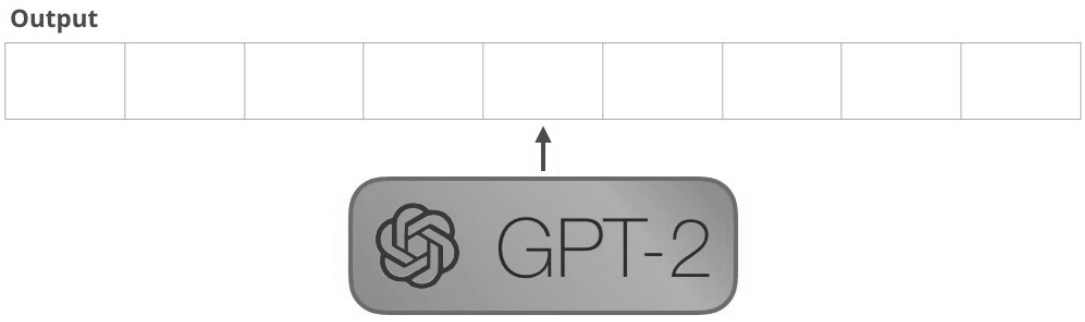



Fig 11 - [2]

These models really operate by adding each token that is created to the input sequence one at a time. And in the model's following phase, the new sequence serves as its input. The concept of "auto-regression" is this. One of the concepts behind RNNs' excessive effectiveness [13] is this one.

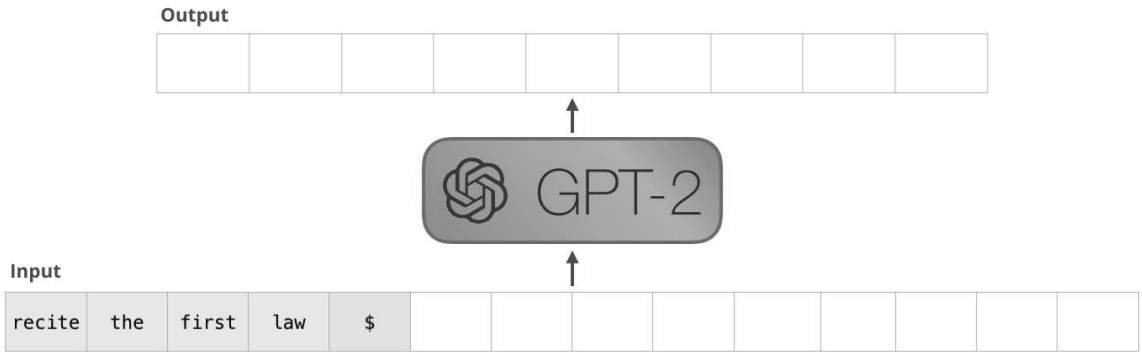



Fig 12 - [2]

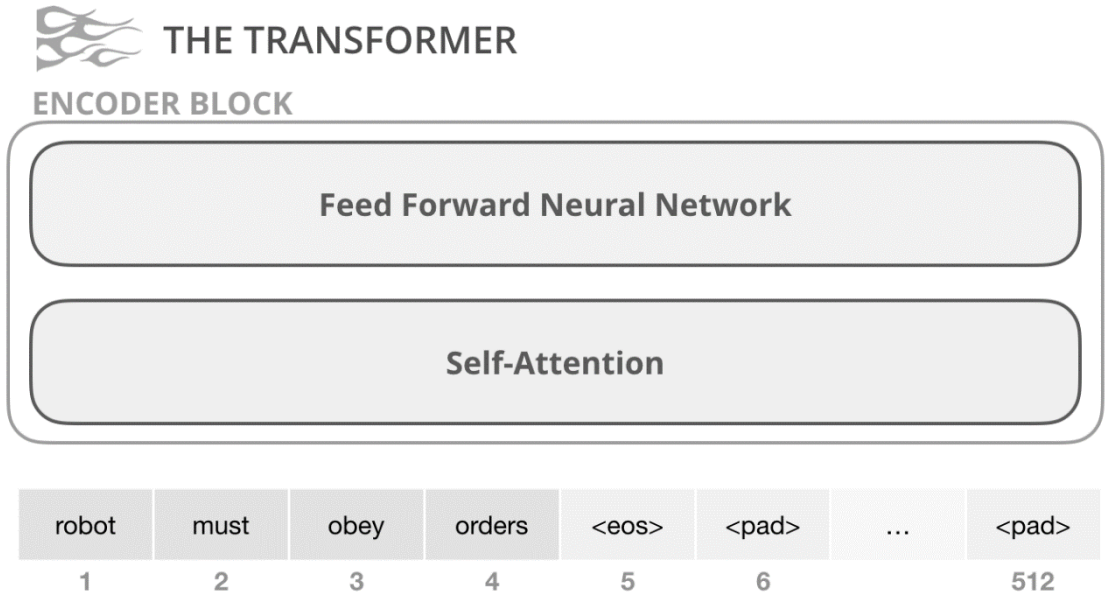
The auto-regressive character of the GPT2 and certain subsequent variants, such TransformerXL and XLNet[28], is evident. It is not BERT. That is a compromise. BERT obtained the capacity to take into account the context on both sides of a word in order to get better outcomes by removing auto-regression. XLNet finds a different technique to include the context on both sides while bringing back autoregression.

### 3.2.1.3. TRANSFORMATION BLOCK DEVELOPMENT

Two different types of transformer blocks were first introduced in the transformer paper [14]:

THE BLOCK ENCODER

The encoder block comes first:




A transformer paper's initial encoder block can accept inputs for up to a predetermined maximum sequence length (for example, 512 tokens). If an input sequence is less than this threshold, it is OK; the remaining characters will simply be padded.

Fig 13 - [2]

THE BLOCK DECODER

Second, there is the decoder block, which differs slightly architecturally from the encoder block by adding a layer that enables it to focus on particular encoder segments:

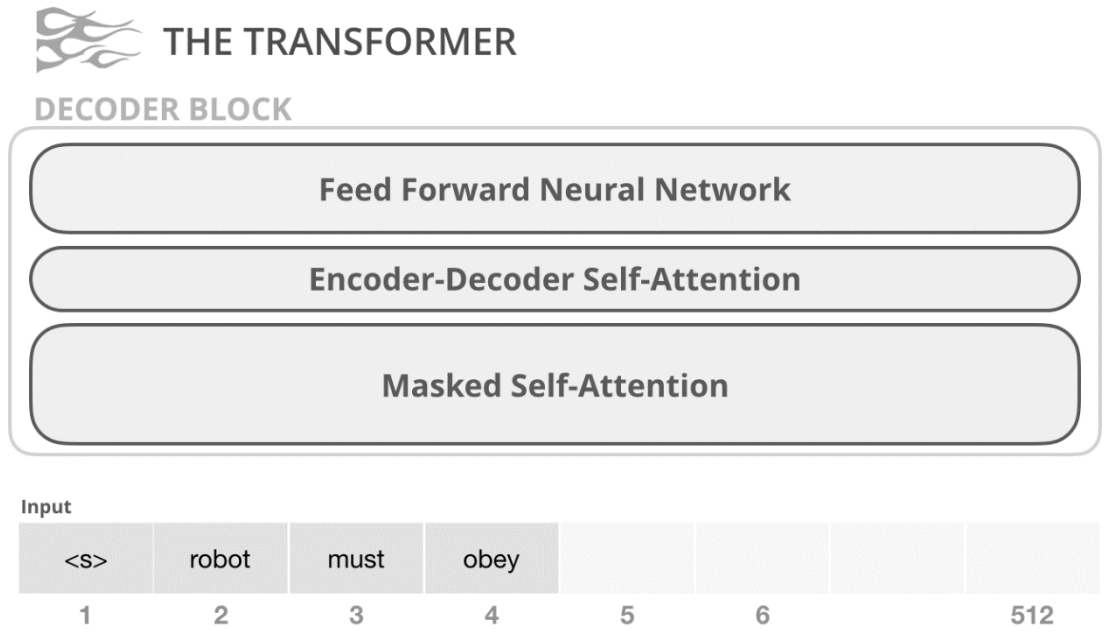



Fig 14 - [2]

One significant distinction is that, unlike BERT, this self-attention layer covers future tokens by interfering with the computation of self-attention and obstructing information from tokens that are to the right of the location being computed.

The route of position #4, for instance, demonstrates that only the present and preceding tokens are permitted to be attended to:

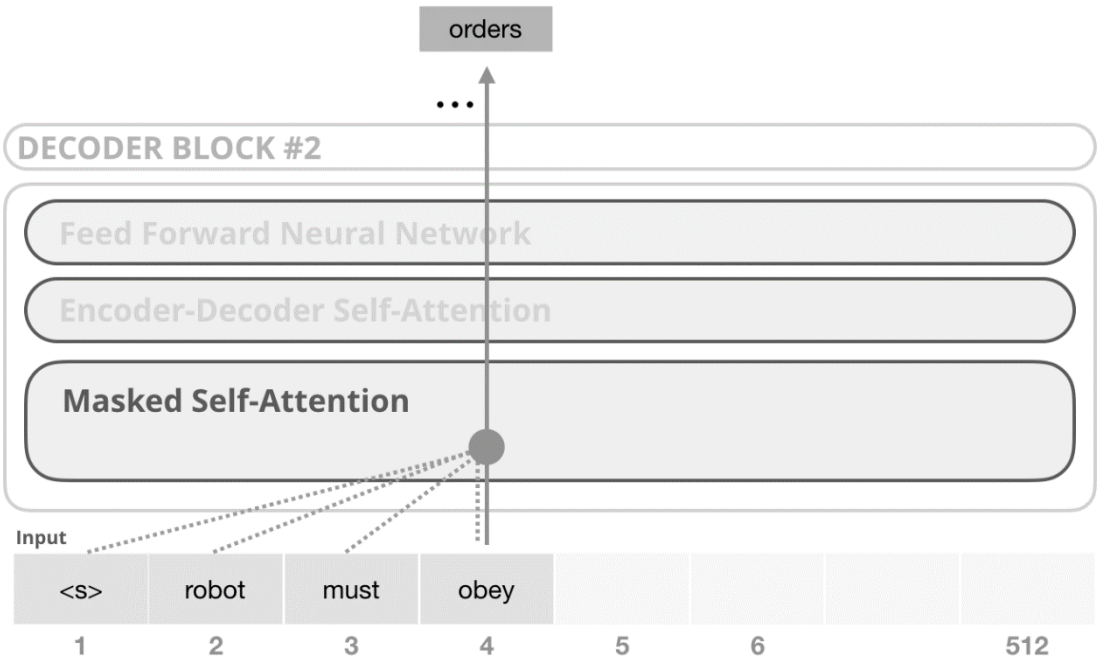



Fig 15 - [2]

It's critical to understand how self-attention (what BERT utilises) and veiled self-attention (what GPT-2 uses) differ from one another. A position may peek at tokens to its right thanks to a typical self-attention block. Self-attention that is concealed prevents that from occurring:

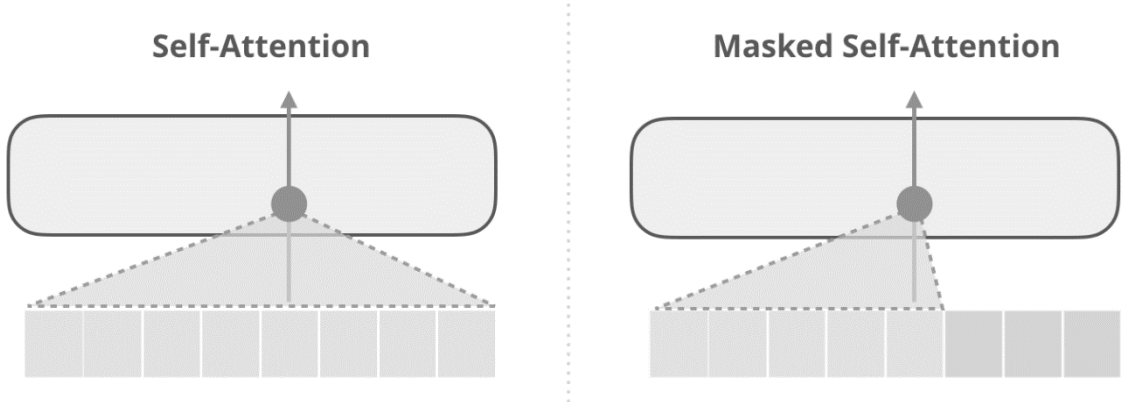
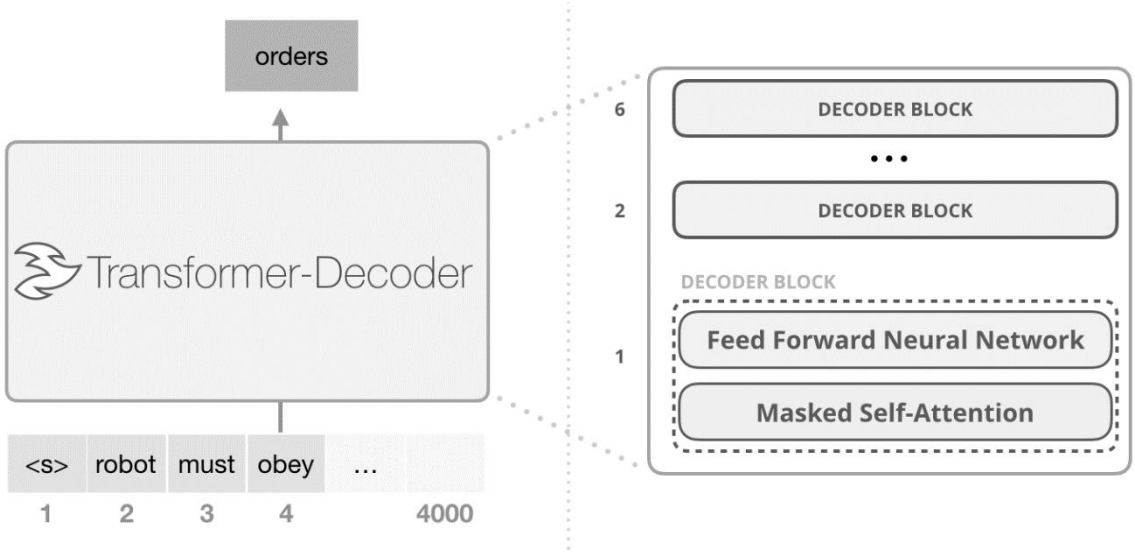



Fig 16 - [2]

THE BLOCK FOR DECODERS

Following the first study, Generating Wikipedia by Summarising Long Sequences [15] suggested a different configuration of the transformer block that may do language modelling. The Transformer encoder was discarded in this variant. Let's call the design "Transformer-Decoder" as a result. A stack of six transformer decoder blocks comprised this early transformer-based language model:




The decoder blocks match one another. You can tell that the first one's self-attention layer is the veiled variation since I have enlarged it. Observe that the model has significantly improved from the 512 tokens in the initial transformer to being able to address up to 4,000 tokens in a specific section.

Fig 17 - [2]

With the exception of eliminating the second self-attention layer, these blocks were quite identical to the original decoder blocks. To develop a language model that predicts one letter or character at a time, a similar architecture was looked at in Character-Level Language Modelling with Deeper Self-Attention [16].

These decoder-only blocks are used by the OpenAI GPT-2 model.

### 3.2.1.4. EVALUATING GPT-2

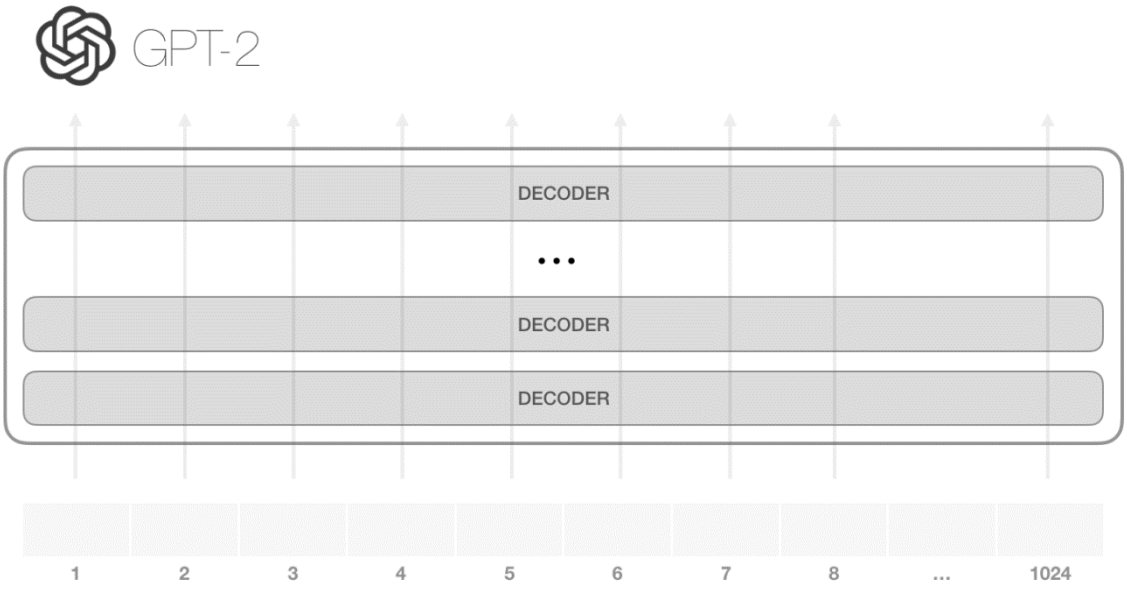



Fig 18 - [2]

Allowing a trained GPT-2 to ramble on its own is the easiest approach to run it; this is known as producing unconditional samples. Alternatively, we may give it instructions to talk about a certain subject (creating interactive conditional samples). In the instance of rambling, we can just give it the start token and tell it to begin producing words (the trained model uses the token |endoftext|> as its start token). Call it "s" instead, please.

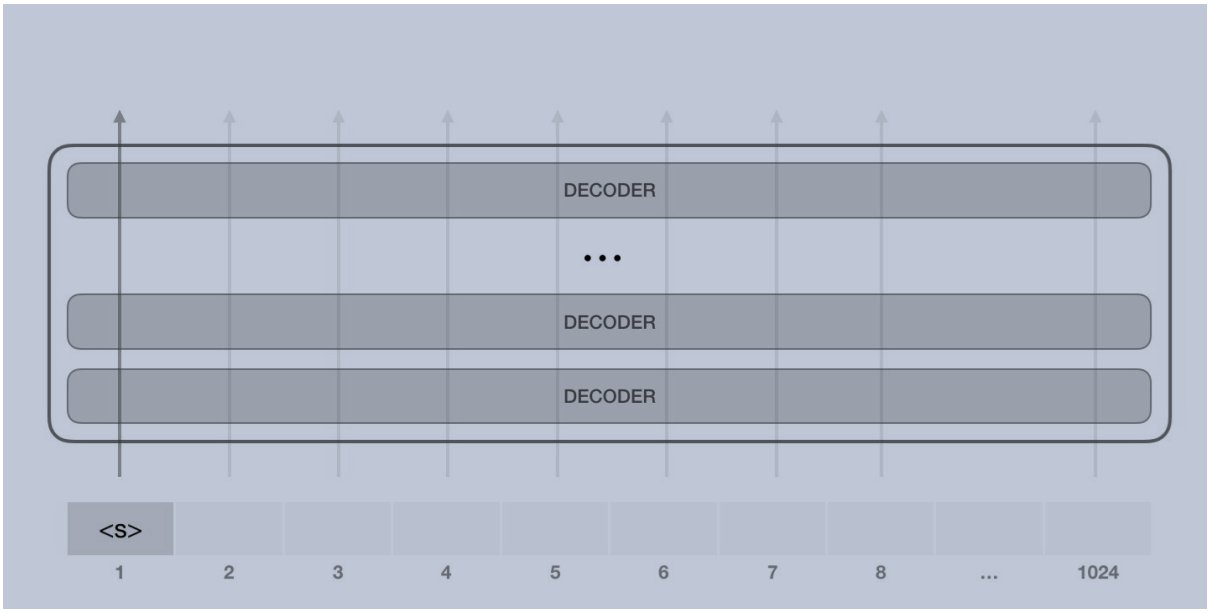



Fig 19 - [2]

Since the model only accepts a single input token, only that path would be operational. The token is processed via each layer in turn, and then a vector is created along that route. The vocabulary of the model (all 50,000 words in the instance of GPT-2) may be used to score that vector. 'the' was the token we chose in this instance since it had the highest probability. However, we can absolutely change things up. For example, if you repeatedly click the recommended word in your keyboard app, it may get stuck in a cycle that cannot be broken without selecting the second or third suggested word. Here, the same is possible. We may instruct the GPT-2 model to sample words other than the top word by setting the top-k parameter to 1 (which instructs the model to do so).

The result from the previous step is added to our input sequence in the following phase, when the model is asked to generate its next prediction:

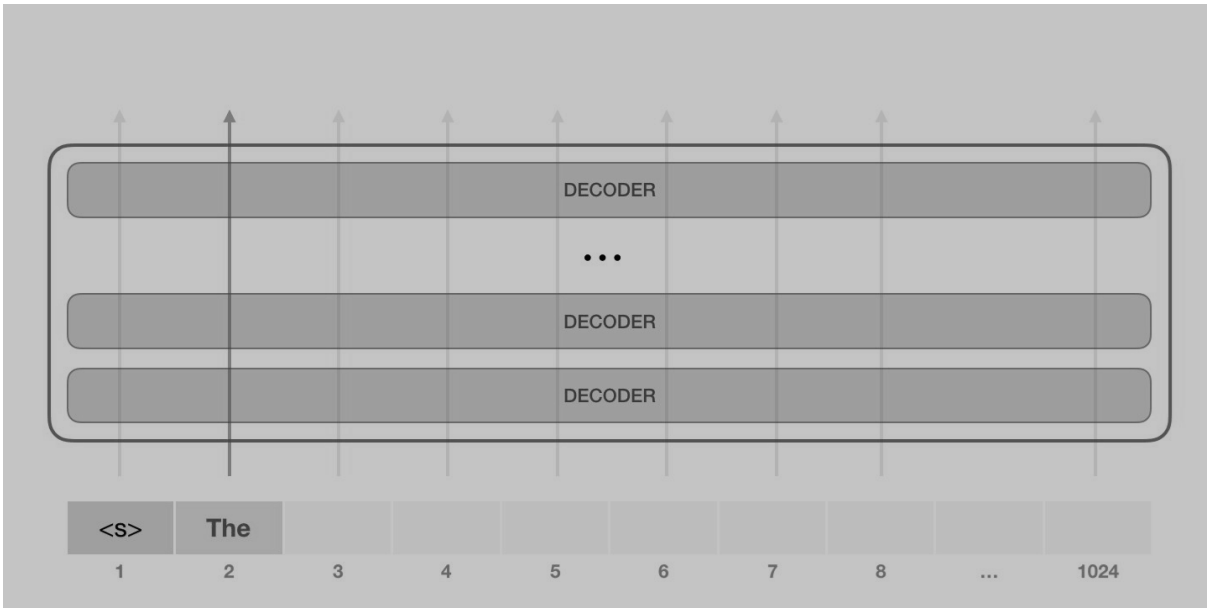



Fig 20 - [2]

Keep in mind that this computation only has the second route active. In order to analyse the second token, each layer of the GPT-2 has preserved its own interpretation of the first token, which it will apply (we'll go into more depth about this in the next section on self-attention). In light of the second token, GPT-2 does not reinterpret the first token.

## 3.2.2. IMPACT OF GPT - 2

GPT-2's Impact in NLP:

- Language Generation: GPT-2 excels in generating realistic and coherent text, making it a significant advancement in the field of natural language generation. Its ability to produce human-like responses has found applications in various areas, including chatbots, content generation, and creative writing.

- Question Answering: GPT-2 has been employed for question answering tasks, where it can generate relevant answers based on given questions. By conditioning the model on the question, it can generate plausible responses, although it may not always provide accurate or factual information.

- Language Translation: GPT-2's powerful language modeling capabilities have been leveraged for machine translation tasks. By conditioning the model on the source language and generating text in the target language, it can generate translations that preserve the contextual information.

- Sentiment Analysis: GPT-2 has been utilized for sentiment analysis tasks, where it can predict the sentiment of a given text. By training the model on labelled datasets, it can learn to classify text as positive, negative, or neutral based on the context and linguistic cues.

- Text Completion: GPT-2 can be used for text completion tasks, where it generates coherent text to complete partially written sentences or paragraphs. This application has been employed in various scenarios, including writing assistance and content generation.

Language Understanding: GPT-2 has demonstrated impressive language understanding capabilities. It can comprehend and generate text that exhibits a nuanced understanding of grammar, context, and semantics. This feature has found applications in tasks such as language modeling, document classification, and information retrieval.

### 3.2.3. CHALLENGES WITH GPT-2:

Challenges with Extractive Text Summarization using GPT-2:

While GPT-2 is a powerful language model, it poses certain challenges when applied to extractive summarization tasks:

- Lack of Control: GPT-2 is a generative model that does not provide fine-grained control over the output. Extractive summarization, on the other hand, requires precise sentence selection based on relevance and importance. GPT-2 may generate summaries that are coherent but not necessarily extractive in nature.

- Information Compression: Extractive summarization aims to condense the essential information from a document into a concise summary. GPT-2, being a generative model, tends to produce text of a similar length to the input, which may not fulfill the goal of compression and conciseness.

- Coherence and Redundancy: GPT-2 tends to generate fluent and coherent text, but it may also introduce redundancy and repetition. In extractive summarization, redundancy is undesirable as it hampers the concise representation of the document's key information.

- Overemphasis on Context: GPT-2's language modeling capabilities are based on contextual information from the training corpus. While this is advantageous for generating coherent text, it may lead to an overemphasis on the immediate context rather than extracting the most salient sentences across the entire document.

### 3.2.4. ADDRESSING OF GPT-2

To leverage GPT-2 for extractive summarization, additional techniques can be employed to select relevant sentences from the generated text. These techniques include:

- Sentence Scoring: Apply scoring mechanisms based on sentence length, importance, or relevance to rank the generated sentences. This allows the extraction of the most relevant sentences from the output.

- Cosine Similarity: Utilize techniques like TF-IDF and cosine similarity to measure the similarity between generated sentences and the source document. Sentences with higher similarity scores can be considered for extraction.

- Supervised Learning: Train a separate model on labelled data with sentence-level annotations for importance. This model can rank and select sentences based on their relevance to the source document, leveraging the generative capabilities of GPT-2 for sentence generation.

- Reinforcement Learning: Apply reinforcement learning techniques to fine-tune GPT-2 for extractive summarization. By defining a reward function based on the quality and relevance of extracted sentences, the model can be trained to generate summaries that align with the extractive summarization objective.

These approaches aim to combine the generative capabilities of GPT-2 with additional techniques to extract salient information effectively. By addressing the challenges specific to extractive summarization, these methods strive to produce concise and coherent summaries.

In conclusion, while GPT-2 is primarily known for its abstractive text generation capabilities, it can still be employed in extractive summarization tasks by integrating it with complementary techniques. GPT-2's impact in NLP has been substantial, with its ability to generate realistic and contextually relevant text opening up numerous possibilities for various language-related tasks.

## 3.3. KL-SUMMARIZER

Another popular approach for extractive summarization is the KL-Sum algorithm, which utilizes Kullback-Leibler (KL) divergence to measure the relevance and importance of sentences. In this article, we will delve into the details of extractive text summarization, specifically focusing on the KL-Summarizer and how it operates. KL-Summarizer, also known as KL-Sum, is an extractive algorithm that utilizes statistical measures to identify important sentences.

The importance of sentences in a given text is determined by the Kullback-Leibler (KL) divergence, which is used by the KL-Sum method. KL divergence is a metric for comparing the differences between two probability distributions. The difference between the probability distribution of the sentences in the document and the probability distribution of sentences in a generic document is quantified in the context of KL-Sum.

### 3.3.1. KL-SUM ALGORITHM

The KL-Sum algorithm is composed of the following key steps: -

Kl Summarizer describes a greedy optimization approach for selecting sentences and ordering them based on a value called "$P_i$." Here is a breakdown of the steps outlined:

1. Initialize an empty set S and set d to 0.
2. Repeat the following loop until the size of S (|S|) reaches a specified limit L:
3. Iterate over a range of values i from 1 to ND (the total number of documents).
4. Calculate a value di using the Kullback-Leibler divergence (KL) between the probability distribution Ps and PD. This value measures the difference between the probability distribution of the selected sentences (Ps) and the probability distribution of the entire document (PD).
5. Add the sentence Si with the minimum di to the set S, and update d to the value of di.
6. If there is no i such that di is less than the current value of d, stop the loop.
7. Finally, the selected sentences in the target document are ordered based on a position index pi, which reflects their position within their respective source documents.
8. For each selected sentence Si extracted from document Dj, compute a position index pi ranging from 0 to 1.
9. The sentences in the target document are then ordered based on the value of pi, following the order of sentences in the source documents.

This strategy seeks to choose the most pertinent sentences from the source documents and arrange them in accordance with where they appear in the source documents [14].

### 3.3.2. ADVANTAGES OF KL - SUMMARIZER

The KL-Summarizer algorithm offers several advantages:

- Extractive Approach: KL-Sum extracts sentence directly from the input document, preserving the original wording and reducing the risk of introducing errors or misinterpretations.

- Simplicity: The algorithm's implementation is relatively straightforward, making it easier to understand and apply.

- Interpretable Results: The sentence scores generated by KL-Sum provide a measure of importance, allowing users to interpret the basis of sentence selection.

### 3.3.3. LIMITATIONS AND ENHANCEMENT OF KL - SUMMARIZER

However, KL-Summarizer also has some limitations:

- Dependency on Generic Document: KL-Sum requires a generic document as a reference to compute sentence scores. The quality and relevance of the generic document can significantly impact the summarization results.

- Lack of Semantic Understanding: KL-Sum does not capture semantic relationships between words or sentences. It relies solely on statistical measures, which may not always reflect the true importance of sentences.

- Sentence Redundancy: KL-Sum may select redundant sentences if they have similar word distributions to important sentences. This redundancy can affect the overall coherence and quality of the summary.

Researchers have proposed several enhancements to address the limitations of KL-Sum and improve the effectiveness of extractive summarization. Some of these include:

- Incorporating Word Embeddings: Word embeddings capture semantic relationships between words, allowing algorithms to better understand the meaning and context of sentences. Integrating word embeddings into KL-Sum could enhance its performance.

- Graph-Based Models: Graph-based models use edge weights to describe the connections between phrases as nodes in a graph. These models can improve coherence and eliminate redundancy in the generated summaries.

- Neural Network-Based Approaches: In text summarization tasks, neural network designs including recurrent neural networks (RNNs) and transformer models have demonstrated promising outcomes. These methods can capture intricate relationships and produce summaries that are more precise.

- Reinforcement Learning: Reinforcement learning techniques can be employed to optimize the summarization process by training models to generate summaries that maximize predefined metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores.

Extractive text summarization is a valuable technique for condensing large amounts of text while preserving important information. The KL-Sum algorithm, which utilizes KL divergence, offers a simple and interpretable approach to extractive summarization. By comparing the word distributions of a document with those of a generic document, KL-Sum assigns scores to sentences and selects the most relevant ones for the summary.

While KL-Sum has its limitations, ongoing research and advancements in NLP and machine learning techniques hold the promise of addressing these challenges and improving the overall effectiveness of extractive text summarization. With further developments, extractive summarization algorithms like KL-Sum can continue to play a significant role in automatically generating concise and informative summaries from large textual data.

## 3.4. LUHN

One popular algorithm for extractive summarization is the LUHN algorithm, which was developed by Hans Peter Luhn in the late 1950s. LUHN's method is based on the assumption that important sentences in a document tend to contain significant and frequent terms. In this article, we will explore the LUHN algorithm in detail, discussing its steps, advantages, limitations, and potential future directions.

The LUHN algorithm is one such approach to extractive text summarization that relies on statistical measures to identify significant sentences.

The LUHN algorithm, proposed by Hans Peter Luhn in 1958, follows a set of steps to generate extractive summaries. It focuses on the frequency of important terms to measure the relevance of sentences.

### 3.4.1. LUHN’s ALGORITHM

The LUHN algorithm's key steps are broken down as follows:

1. In the initial stage, our goal is to identify the words that hold greater significance in conveying the meaning of a document. According to Luhn's approach, this is achieved through a frequency analysis, followed by the identification of important words that are not considered unimportant English words. For example,

| SENTENCES/WORDS | INTERN | OPENGENUS | DEVELOPER | ML |
|---|---|---|---|---|
| An intern at OpenGenus | 1 | 1 | 0 | 0 |
| Developer at OpenGenus | 0 | 1 | 1 | 0 |
| A ML intern | 1 | 0 | 0 | 1 |
| An ML developer | 0 | 0 | 1 | 1 |

Fig 21 - [3]

We can observe from the above table that stop words like a and an are not taken into account while evaluating.

2. Moving on to the second phase, we determine the most frequently occurring words in the document. From this set, we select a subset that excludes the commonly used English words but still retains words of importance. This phase typically involves three steps:

   i. The process begins by converting the sentences' content into a mathematical expression or vector, often represented as a binary representation. To accomplish this, we utilize a bag-of-words approach that disregard filler words. Filler words are typically auxiliary words that do not significantly contribute to the document's meaning. Then, we tally up all the valuable words that remain.

   ii. In this step, we access sentences using a sentence scoring technique. One possible scoring method, as demonstrated below, involves calculating a score based on the number of meaningful words squared divided by the span of those meaningful words. Here, the span refers to the portion of the sentence (or document) that contains all the meaningful words. Additionally, tf-idf can be employed to prioritize words within a sentence based on their rarity across a broader corpus.

iii. Once the sentence scoring process is complete, the final step involves selecting sentences with the highest overall rankings.

To summarize, Luhn's algorithm for text summarization entails determining the significance of words in the document, evaluating sentences based on their meaningful word count and span, and ultimately selecting the sentences with the highest scores to form the summary [2][18].

### 3.4.2. ADVANTAGES OF LUHN ALGORITHM

The LUHN algorithm offers several advantages:

- Simplicity: The LUHN algorithm is relatively simple to implement and understand, making it accessible for various applications.

- Extractive Approach: By selecting sentences directly from the original document, the LUHN algorithm preserves the exact wording and reduces the risk of introducing errors or misinterpretations.

- Computational Efficiency: The algorithm's straightforward nature makes it computationally efficient, allowing it to handle large volumes of text with reasonable processing times.

### 3.4.3. LIMITATIONS AND ENHANCEMENTS OF LUHN ALGORITHM

However, the LUHN algorithm also has some limitations:

- Lack of Semantic Understanding: The algorithm solely relies on statistical measures and term frequency. It does not capture the semantic relationships between words or sentences, potentially leading to the inclusion of less relevant sentences in the summary.

- Sentence Redundancy: The LUHN algorithm may select multiple sentences conveying similar information if they contain the same or similar important terms. This redundancy can affect the overall coherence and quality of the summary.

- Sensitivity to Term Frequency: The LUHN algorithm heavily relies on term frequency as a measure of importance. This sensitivity can lead to the inclusion of sentences that may be less informative but contain frequently occurring terms.

Researchers have proposed several enhancements and variations to the LUHN algorithm to address its limitations and improve the effectiveness of extractive summarization. Some of these include:

- Word Embeddings: Integrating word embeddings, such as Word2Vec or GloVe, into the LUHN algorithm can enhance its performance by capturing semantic relationships between words. Word embeddings provide a more nuanced representation of word meanings and can improve the selection of important terms and sentences.

- Graph-Based Models: The relevance of sentences is determined using graph algorithms, which are used in graph-based models like LexRank and TextRank that represent sentences as nodes in a graph. These models can assist in reducing repetition and enhancing the summary's coherence.

- Neural Network-Based Approaches: In text summarization tasks, neural network designs including recurrent neural networks (RNNs) and transformer models have demonstrated promising outcomes. These methods may capture intricate relationships and semantic data, allowing for more precise and thorough summaries.

- Evaluation Metrics: The quality of the produced summaries may be evaluated objectively by creating more reliable assessment measures for extractive summarization. It is possible to leverage established metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and develop new metrics to measure qualities like coherence and readability.

The LUHN algorithm, proposed by Hans Peter Luhn, is a popular extractive text summarization method that relies on term frequency to identify important sentences in a document. By assigning weights to terms and scoring sentences based on the presence of these terms, the LUHN algorithm generates extractive summaries.

While the LUHN algorithm has its limitations, ongoing research and advancements in NLP and machine learning techniques offer promising avenues for improvement. By incorporating semantic understanding, graph-based models, and neural network-based approaches, extractive summarization algorithms like LUHN can enhance their effectiveness and produce more coherent and informative summaries. With further developments, extractive summarization techniques will continue to play a vital role in automatically generating concise and relevant summaries from large textual data.

## 3.5. LEX

LexRank is one of the popular algorithms used for extractive summarization. It leverages graph-based ranking methods to identify key sentences based on their similarity and importance within the document. In this article, we will explore LexRank in detail, discussing its steps, advantages, limitations, and potential future directions.

The critical task of text summarization in natural language processing (NLP) is reducing the amount of information in a document while maintaining its vital content. In order to provide a summary, extractive summarising techniques choose and extract pertinent lines or phrases from the original text. This strategy is favoured since it may preserve the original phrasing and lowers the possibility of adding mistakes or misinterpretations.

LexRank is an extractive summarization algorithm that utilizes graph-based ranking methods to identify important sentences within a document.

The LexRank algorithm was introduced by Erkan and Radev in 2004. It draws inspiration from the PageRank algorithm, which is used by search engines to rank web pages based on their importance. LexRank applies similar principles to rank sentences within a document.

### 3.5.1. LEX ALGORITHM

Here's an overview of the main steps involved in the LexRank algorithm:

1. Pre-processing: Pre-processing operations on the input document include stemming, stop word removal, and tokenization. The paper may be broken down into individual phrases using these processes, and unnecessary words can be eliminated.

2. Similarity Matrix Construction: The bag of words model, where N is the total number of words in a particular language, is used to characterise N-dimensional vectors in order to determine similarity. The idf of the word multiplied by the number of times it appears in the phrase determines the value of the related dimension in the sentence's vector representation.

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}}$$

(Eq.1)

The number of times the word w appears in the sentence s is shown by the symbol $\text{tf}_{w,s}$.

3. Graph Construction: The similarity matrix is converted into a graph representation, where phrases are represented as nodes, and relationships between them are shown as edges. In order to create a weighted graph, each sentence is linked to other sentences that it is comparable to.

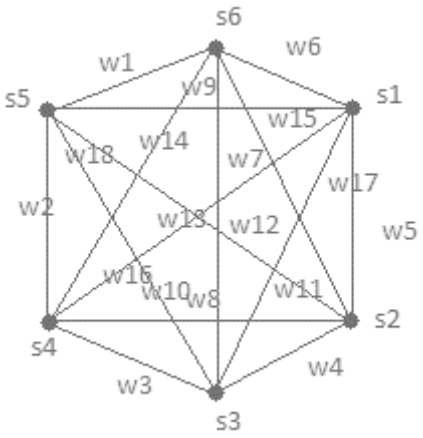



Fig 22 – [4]

Where $S_i$ corresponds to the sentences at each of the vertices and $W_{ij}$ to the weights along the edges.

4. PageRank Calculation: The PageRank algorithm is applied to the graph to rank the sentences based on their importance. PageRank assigns higher scores to sentences that are connected to other sentences with high similarity scores.

5. Sentence Selection: Sentences with the highest PageRank scores are selected to form the summary. The number of sentences included in the summary can be predefined or based on a desired summary length.

6. Summary Generation: The selected sentences are combined to create the final summary, preserving the original order as they appeared in the document[4].

### 3.5.2. ADVANTAGES OF LEXRANK

The LexRank algorithm offers several advantages:

- Graph-Based Approach: LexRank utilizes a graph-based representation to capture the relationships and similarities between sentences. This approach helps improve the coherence and eliminate redundancy in the generated summaries.

- Importance-Based Ranking: The algorithm assigns importance scores to sentences based on their similarity to other sentences. This ranking ensures that the most relevant and representative sentences are selected for the summary.

- Language Independence: LexRank can be applied to documents in various languages without requiring language-specific modifications. It relies on the inherent structure of the text and does not rely on language-specific linguistic features.

### 3.5.3. LIMITATIONS AND ENHANCEMENTS OF LEXRANK

However, LexRank also has some limitations:

- Lack of Semantic Understanding: While LexRank captures similarity between sentences based on their textual content, it does not capture semantic relationships or deeper meaning. As a result, it may include sentences that are similar but convey different information.

- Sensitivity to Similarity Measure: The choice of similarity measure used to construct the similarity matrix can impact the performance of LexRank. Different similarity measures may yield different results, and selecting an appropriate measure is crucial.

- Computationally Intensive: The LexRank algorithm involves the construction and manipulation of a similarity matrix and graph, which can be computationally intensive for large documents. However, various optimizations and approximations can be applied to mitigate this issue.

Researchers have proposed several enhancements and variations to the LexRank algorithm to address its limitations and further improve extractive summarization. Some of these include:

- Extended Context: Incorporating a wider context by considering not only sentence-level similarity but also document-level information can improve the selection of important sentences. This can be achieved by incorporating document-level features or by using context-aware embeddings.

- Neural Network-Based Approaches: Neural network architectures, such as recurrent neural networks (RNNs) and transformer models, have shown promising results in text summarization tasks. These approaches can capture complex dependencies and semantic information, enabling more accurate and coherent summaries.

- Domain-Specific Modifications: Adapting the LexRank algorithm to specific domains or genres of text can enhance its performance. Domain-specific modifications can include incorporating domain-specific features, adjusting similarity measures, or fine-tuning the ranking process based on domain-specific criteria.

- Multi-Document Summarization: Extending LexRank to handle multiple documents can enable summarization of collections of related texts. Techniques such as cross-document similarity calculation and clustering can be employed to generate summaries that capture the key information from multiple sources.

LexRank is a graph-based algorithm that ranks sentences within a document based on their similarity and importance. By leveraging graph-based ranking methods, LexRank generates extractive summaries by selecting sentences with high importance scores.

While LexRank has its limitations, ongoing research and advancements in NLP and machine learning techniques offer potential solutions. By incorporating semantic understanding,

exploring different similarity measures, and adapting the algorithm to specific domains, extractive summarization algorithms like LexRank can continue to improve their effectiveness and generate more coherent and informative summaries. With further developments, extractive summarization techniques will play a vital role in automatically generating concise and relevant summaries from large textual data.

## 3.6. WORD RANK

Word Rank is an extractive text summarising algorithm that chooses sentences that include key phrases from a document and uses them to build a summary. Unlike other algorithms that focus on sentence-level analysis, Word Rank operates at the word level. In this article, we will explore Word Rank in detail, discussing its steps, advantages, limitations, and potential future directions.

Word Rank is an extractive summarization algorithm that operates by identifying significant terms and selecting sentences containing those terms.

The Word Rank algorithm, introduced by Aliaksei Severyn and Alessandro Moschitti in 2015, focuses on the importance of terms within a document. Word Rank establishes the importance of sentences by giving words weights depending on their frequency and placement in the text.

### 3.6.1. WORD RANK ALGORITHM

Here is an overview of the main steps involved in the Word Rank algorithm:

1. Pre-processing: The initial document undergoes several pre-processing procedures to enhance its suitability for further analysis. These steps involve tokenization, the elimination of stop words, and stemming. Through tokenization, the document is fragmented into individual sentences or smaller units, facilitating a more detailed examination. Stop words, which are inconsequential words that do not carry substantial meaning, are then eliminated from the document. This removal aids in focusing on the more essential content that contributes significantly to the summary. Lastly, stemming is applied to reduce words to their root form, enabling consolidation of related words and reducing redundancy. Overall, these pre-processing steps work together to break down the input document, eliminate

unnecessary words, and extract the core information that holds relevance for generating a concise summary.

2. Term Frequency Calculation: In order to determine the occurrence of each term in the document, the frequency of each term is computed. This process entails tallying the number of times each term appears within the document. The terms considered can range from individual words to phrases, depending on the desired level of detail and specificity. By performing this frequency calculation, we obtain a quantitative representation of how often each term occurs, enabling further analysis and insights into the document's content. The purpose of this step is to provide a clear understanding of the distribution and prominence of different terms within the document, helping to identify significant patterns or key elements that contribute to its overall meaning.

3. Term Position Weighting: When assessing the importance of each term within the document, special consideration is given to its position, and a weighting scheme is applied. This scheme grants higher weights to terms that appear at the beginning of sentences. The underlying assumption is that terms occurring early in sentences carry more significance and serve as better representatives of the content. By assigning these higher weights to such terms, the weighting scheme acknowledges their potential impact on conveying essential information and conveying the core message of the document. This approach aims to capture the relative importance of terms based on their placement within sentences, emphasizing the belief that terms appearing earlier hold greater relevance and contribute more significantly to the overall meaning. Thus, by considering the position of each term and implementing this weighting scheme, we can effectively highlight and prioritize the terms that are likely to be more crucial in understanding the document's content.

4. Sentence Scoring: To determine the significance of sentences within the document, a scoring process is employed, which relies on the weights assigned to the terms they contain. The scores for each sentence are computed by adding up the weights of the terms present within that particular sentence. Consequently, sentences with higher scores are deemed more important and are chosen to be included in the summary. This scoring mechanism enables the identification of sentences that encompass the most relevant and crucial information by considering the cumulative weight of the terms they contain. By selecting sentences with higher scores, the summary can effectively capture the key points and

essential content from the document, prioritizing those sentences that carry more weight and contribute significantly to conveying its overall meaning.

5. Sentence Selection: The summary is constructed by choosing sentences with the most elevated scores. The selection process entails identifying sentences that have achieved the highest scores and including them in the summary. The number of sentences to be included in the summary can be determined in advance or determined based on the desired length of the summary. This approach ensures that the most important and informative sentences are incorporated, as they have attained the highest scores through the scoring mechanism. By curating the summary from these highly scored sentences, a concise and representative overview of the document can be generated. The flexibility of determining the number of sentences in the summary allows for customization based on specific requirements, such as the desired level of detail or the designated length for the summary output.

6. Summary Generation: The final summary is generated by merging the selected sentences, preserving their original order as they appeared in the document. This process involves combining the chosen sentences in a cohesive manner, ensuring that their sequence is maintained to reflect the original flow of information. By adhering to the original order, the summary retains the logical progression and coherence present in the source document. This approach aims to provide a concise representation of the document's key points while maintaining the contextual integrity of the information. By combining the selected sentences in their original arrangement, the final summary effectively captures the essence of the document and presents it in a condensed form that aligns with the structure and coherence of the original content[19][20].

### 3.6.2. ADVANTAGES OF WORD RANK

The Word Rank algorithm offers several advantages:

- Term-Level Analysis: Word Rank operates at the word level, allowing for a more granular analysis of the document. By focusing on individual terms and their positions, the algorithm captures important information that might be missed by sentence-level approaches.

- Simplicity: The algorithm's implementation is relatively simple, making it easy to understand and apply. The straightforward nature of Word Rank enables efficient processing of large volumes of text.

- Extractive Approach: Word Rank directly selects sentences from the original document, preserving the original wording and reducing the risk of introducing errors or misinterpretations.

### 3.6.3. LIMITATIONS AND ENHANCEMENT OF WORD RANK

However, Word Rank also has some limitations:

Lack of Semantic Understanding: The algorithm does not capture semantic relationships between words or sentences. It relies solely on statistical measures, which may not always reflect the true importance or relevance of sentences.

Sensitivity to Term Frequency and Position: Word Rank heavily relies on term frequency and position in the document. While these factors can indicate importance, they may not always align with the overall content significance.

Sentence Redundancy: Word Rank may select multiple sentences that convey similar information if they contain the same or similar important terms. This redundancy can impact the coherence and quality of the summary.

Researchers continue to explore enhancements and variations of the Word Rank algorithm to address its limitations and improve extractive summarization. Some potential directions for improvement include:

- Semantic Analysis: Integrating semantic understanding into the Word Rank algorithm can enhance the selection of relevant sentences. Techniques such as word embeddings or semantic similarity measures can be employed to capture semantic relationships and improve the accuracy of term importance estimation.

- Contextual Information: Incorporating contextual information, such as document-level or discourse-level features, can provide a more comprehensive view of the text. This additional information can help identify key terms and sentences that contribute to the overall meaning and coherence of the document.

- Hybrid Approaches: Combining Word Rank with other extractive summarization algorithms or techniques can leverage the strengths of different methods. For example, incorporating graph-based methods or neural network-based approaches can improve the overall performance and effectiveness of the summarization process.

- Evaluation Metrics: Developing robust evaluation metrics specific to extractive summarization can enable better assessment and comparison of the quality of generated summaries. Traditional metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) can be utilized, along with new metrics that capture aspects like coherence, informativeness, and readability.

Word Rank is an extractive summarization algorithm that selects sentences based on the importance of terms within a document. By assigning weights to terms and considering their frequency and position, Word Rank generates extractive summaries that capture key information.

While Word Rank has its limitations, ongoing research and advancements in NLP and machine learning offer promising avenues for improvement. By incorporating semantic understanding, contextual information, and hybrid approaches, extractive summarization algorithms like Word Rank can enhance their effectiveness and produce more coherent and informative summaries. With further developments, extractive summarization techniques will continue to play a significant role in automatically generating concise and relevant summaries from large textual data.

## 3.7. ROUGE SCORE

A popular metric for assessing the effectiveness of automatic summarization systems is the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score. The process of creating

succinct and insightful summaries of lengthy materials, such as articles, papers, or conversations, is known as automatic summarising.

Automatic summary aims to extract the essential details from the source material and present them in a streamlined way so that consumers may quickly understand the major ideas without having to read the full thing. ROUGE scores serve as a quantitative evaluation of the efficacy of the summarising method by measuring how similar the generated summary is to one or more reference summaries.

ROUGE scores are typically calculated at different levels, such as ROUGE-N and ROUGE-L. Let's explore these in more detail:

ROUGE-N: This ROUGE version counts the number of n-grams that overlap between the reference summary and the produced summary. A continuous run of n words is known as an n-gram. Based on the matching n-grams between the two summaries, ROUGE-N determines the accuracy, recall, and F-measure. Frequently used answers for n are 1 (unigrams), 2 (bigrams), or 3 (trigrams).

ROUGE-L: The longest common subsequence (LCS) between the produced summary and the reference summary is calculated by ROUGE-L. Regardless of the word arrangement, the LCS is the longest string of words to occur in either summary. ROUGE-L calculates precision, recall, and F-Measureby taking into consideration the length of the LCS as well as the lengths of the produced and reference summaries.

The calculation of ROUGE scores involves several steps:

- Tokenization: Both the produced and the reference summaries are tokenized into discrete words or units. Depending on the required degree of analysis, the text is divided into smaller components called tokens, such as words, phrases, or paragraphs.

- N-gram extraction: N-grams are extracted from both the reference and generated summaries based on the chosen value of n. For example, if we consider bigrams (n=2), the summaries are split into sequences of two adjacent words. The n-gram extraction captures the local word order information.

- Overlap computation: It is determined how many n-grams in the reference and produced summaries coincide. This is done for ROUGE-N by calculating the percentage of n-grams from the produced summary that match n-grams in the reference summary.

- Counting: Recall (R) and precision (P) are calculated by comparing the number of overlapping n-grams to the total n-grams in the reference summary and the total n-grams in the produced summary, respectively. Precision measures the amount of created information that is pertinent to the reference summary, whereas recall measures the amount of pertinent information from the reference summary that is recorded by the generated summary.

- F-measure: The F-Measure provides a fair assessment of the performance of the summarization system by combining recall and precision into one score. F = (2 * P * R) / (P + R) is the formula used to compute the harmonic mean of recall and accuracy. The F-Measure encourages a balance between the two by accounting for situations in which accuracy and recall may have distinct values.

- Aggregation: Different ROUGE scores, such as ROUGE-1, ROUGE-2, or ROUGE-L, can be computed. To obtain an overall evaluation, the F-measures for each level can be averaged or combined using a weighted average. The specific aggregation method may depend on the evaluation requirements or preferences.

The similarity between produced and reference summaries is quantified by ROUGE scores. Greater content overlap and better alignment between the produced and reference summaries are both indicated by higher ROUGE ratings. These results enable researchers and developers to evaluate their models, compare various summarising systems, and monitor the advancement of autonomous summarization research.

It is important to note that ROUGE scores have limitations. They rely on the availability of reference summaries, which may not always be present or may introduce bias if they are generated by humans. Additionally, ROUGE scores primarily focus on lexical overlap and do not capture other important aspects of summary quality, such as coherence, fluency, or the ability to capture salient information.

Nevertheless, ROUGE scores serve as a valuable tool in the evaluation and development of automatic summarization systems. They provide a standardized and objective measure of summary quality, facilitating the advancement of research in this field and enabling the comparison of different approaches and techniques [21].

### 3.7.1. PRECISION

The percentage of correctly predicted positive outcomes (true positives) is a measure of precision. Its formulation is:

$$\frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}} = \frac{\text{No.of Correctly Predicted Postive Instances}}{\text{No.of Total Positive Predicted you made}} = \frac{\text{No.Of Correctly Predicted People with Cancer}}{\text{No,of People you Predicted to have cancer}} \quad \text{(Eq. 2)}$$

### 3.7.2. RECALL

Recall is a measurement of the proportion of positive cases in the data that the classifier correctly predicted out of all of them. At times, it is also referred to as sensitivity. The following is its formula:

$$\frac{\text{True Posititves}}{\text{True Positives}+\text{False Negatives}} = \frac{\text{No.Of Correctly Predicted Positive Instance}}{\text{No,of Total Positive Instance in the Dataset}} = \frac{\text{No.of Correctly Predicted People with Cancer}}{\text{No.of People with Cancer in the Dataset}} \quad \text{(Eq. 3)}$$

### 3.7.3. F1-MEASURE

A measurement that combines recall and accuracy is the F1-Measure. It is commonly referred to as the harmonic mean of the two. The harmonic mean is another way to calculate a "average" of numbers; it is often seen to be better suited for ratios than the traditional arithmetic mean (such as recall and accuracy). The F-Measure computation in this case is as follows:

$$\text{F-MEASURE} = 2 * \frac{\text{PRECISION}+\text{RECALL}}{\text{PRECISION}*\text{RECALL}} \quad \text{(Eq. 4)}$$

## 3.8. BERT SCORE

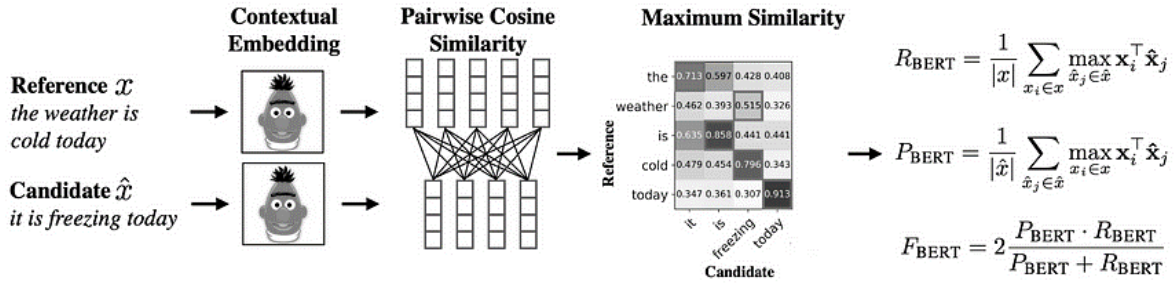



Fig 23 – [5]

A statistic known as BERTScore is used to assess the quality of produced text or translations by contrasting them with reference material. It is based on Google's state-of-the-art language model, the Bidirectional Encoder Representations from Transformers (BERT) model. To determine how comparable the generated and reference texts are, BERTScore makes use of the contextualised word embeddings generated by BERT.

To understand BERTScore in more detail, let's explore the following aspects:

Contextualized Word Embeddings: BERT, as a transformer-based model, produces contextualized word embeddings. Unlike traditional word embeddings that represent words in a static manner, contextualized embeddings capture the meaning of words within the context of the sentence. BERT achieves this by considering both left and right context during the training process. These embeddings are more informative and can better represent the nuances and meaning of words.

Similarity Calculation: BERTScore uses contextualized word embeddings to compute the similarity between the generated and reference text. Instead of relying solely on lexical or surface-level matching, BERTScore takes into account the contextual information encoded in the embeddings. This allows for a more nuanced assessment of the similarity between the two texts.

Sentence Segmentation: Before computing the BERTScore, the generated and reference texts are segmented into sentences or smaller units. This step ensures that the comparison is performed at a more granular level, capturing the similarity between corresponding segments of the two texts.

Word Alignment: BERTScore calculates a word alignment matrix that captures the similarity between individual words in the generated and reference text. The alignment matrix is computed by comparing the contextualized word embeddings of each word pair. The alignment matrix reflects how well each word in the generated text aligns with the words in the reference text.

Greedy Matching: To determine the best alignment between the words in the generated and reference text, BERTScore employs a greedy matching algorithm. This algorithm maximizes the similarity between the aligned words while ensuring that each word is aligned only once. It starts by finding the highest similarity score and then moves to the next highest until all words are aligned.

Precision, Recall, and F1-Measure: Based on the word alignment matrix, BERTScore computes precision, recall, and F1-Measure. Precision measures the proportion of generated words that align well with the reference words, while recall measures the proportion of reference words that align well with the generated words. F-Measure is the harmonic mean of precision and recall, providing a balanced evaluation metric.

BERTScore Calculation: BERTScore aggregates the precision, recall, and F-Measure across all sentences or units to obtain an overall BERTScore. This aggregated score provides a comprehensive measure of the similarity between the generated and reference text.

Tokenization and Subword Units: BERTScore handles tokenization and subword units to ensure consistency and accuracy in the comparison process. It uses the same tokenization methods as BERT to ensure that the generated and reference text are aligned at the subword level.

BERTScore offers several advantages over traditional evaluation metrics. Firstly, it leverages the power of contextualized embeddings to capture the semantic similarity between texts. This

allows for a more fine-grained and accurate evaluation, as it considers the overall meaning rather than relying solely on lexical overlap. Secondly, BERTScore can handle variations in sentence length and structure, making it suitable for evaluating texts of different lengths. Finally, BERTScore aligns words between the generated and reference text, providing insights into specific word correspondences and mismatches.

BERTScore has gained popularity in various natural language processing tasks, including machine translation, text summarization, and text generation. Its use in evaluating the quality of generated text has become increasingly common, as it offers a robust and context-aware metric that aligns with human judgments.

In summary, BERTScore is a metric that utilizes contextualized word embeddings from the BERT model to compute the similarity between generated and reference text. By considering the context and leveraging alignment techniques, BERTScore provides a comprehensive evaluation of the quality of generated text, enabling researchers and developers to assess and compare the performance of their models accurately [22].

## 3.9. MOVER SCORE

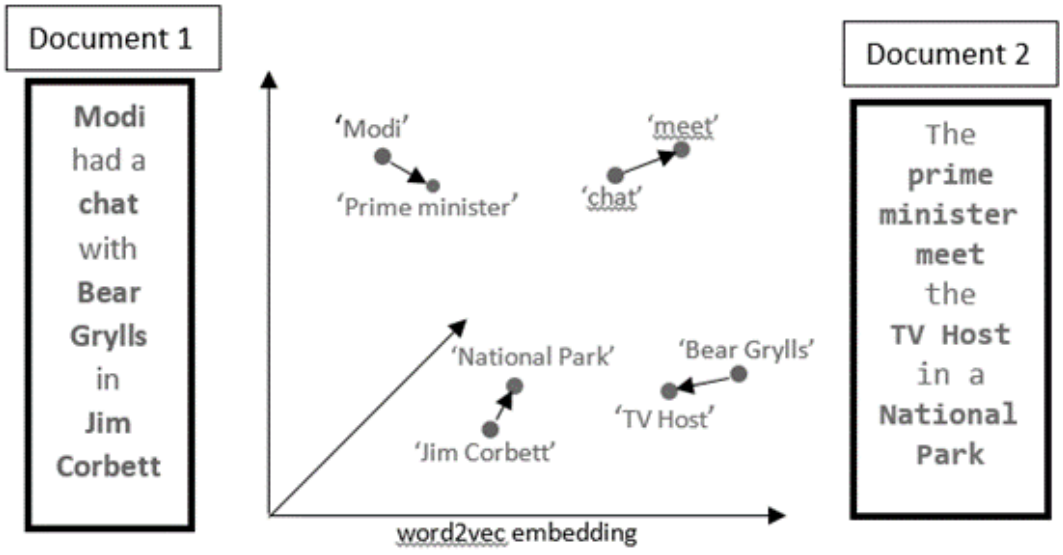



Fig 24 – [6]

The MoverScore statistic compares produced text or translations against reference material to assess the quality of each. It is based on the idea of "Word Mover's Distance" (WMD), which computes the least amount of work necessary to change the word embeddings of one text document into another to determine how different two text documents are. MoverScore extends the idea of WMD to provide a more fine-grained and robust evaluation of generated text.

To understand MoverScore in more detail, we must explore the following aspects:

Word Mover's Distance (WMD): Word Mover's Distance is a measure of semantic similarity between two text documents. It quantifies the "distance" between two sets of word embeddings by computing the minimum amount of "movement" or "effort" required to transform the word embeddings of one document into the other. The WMD considers the semantic meaning of words based on their contextualized embeddings and captures the overall similarity between documents beyond surface-level matching.

Contextualized Word Embeddings: Similar to BERTScore, MoverScore relies on contextualized word embeddings to capture the meaning of words in the context of the document. These embeddings are produced by models such as BERT or GPT, which take into account the surrounding context to generate word representations. Contextualized word embeddings allow for a more nuanced and accurate assessment of the similarity between words and documents.

Word Embedding Similarity: To compute the WMD, MoverScore utilizes the similarity between word embeddings. The similarity can be measured using different distance metrics, such as Euclidean distance or cosine similarity. The choice of similarity metric depends on the specific implementation and requirements of the evaluation.

Word Embedding Space: MoverScore operates in the space of word embeddings, where each word is represented as a vector. These word embeddings are learned during the training of models like BERT or GPT, and they capture the semantic information of words based on the context in which they appear. By leveraging this embedding space, MoverScore can measure the similarity between words and documents using geometric distances.

Sentence Segmentation: Similar to other evaluation metrics, MoverScore segments the generated and reference text into sentences or smaller units. This segmentation allows for a more granular comparison and evaluation of corresponding segments between the two texts. It ensures that the evaluation captures the quality of the generated text at a more local level, considering the coherence and fluency of individual sentences.

Word Alignment and Transport Plan: MoverScore aligns words between the generated and reference text to compute the Word Mover's Distance. It finds the optimal assignment of words from the generated text to words in the reference text based on their semantic similarity. The alignment process produces a transport plan, which represents how words from the generated text should be "transported" to match the words in the reference text. The transport plan indicates the movement required to transform the generated text into the reference text, forming the basis of the MoverScore calculation.

MoverScore Calculation: MoverScore calculates the Word Mover's Distance by considering the transport plan and the distances between the aligned word embeddings. The WMD represents the minimum "effort" or "distance" required to transform the word embeddings of the generated text into the reference text. MoverScore then converts the WMD into a normalized score by scaling it based on the lengths of the generated and reference texts. The resulting MoverScore provides a measure of the dissimilarity between the generated and reference text, where a lower score indicates a higher level of similarity.

Sentence and Document-level Aggregation: MoverScore can be computed at both the sentence and document levels. At the sentence level, MoverScore calculates the WMD and aggregates the scores across all sentences to obtain an overall MoverScore for the entire document. At the document level, MoverScore takes into account the lengths of the generated and reference texts to ensure fairness in the evaluation. Aggregating scores at different levels provides insights into the quality of the generated text at varying granularities.

MoverScore offers several advantages in evaluating the quality of generated text. Firstly, it captures semantic similarity beyond surface-level matching, as it considers the overall meaning and context encoded in word embeddings. This allows for a more accurate assessment of the semantic quality and coherence of the generated text. Secondly, MoverScore considers the alignment and movement of words, providing insights into how well the generated text matches

the reference text in terms of word choices and semantics. Thirdly, MoverScore is versatile and can be applied to various natural language processing tasks, such as machine translation, text summarization, or text generation.

In summary, MoverScore is a metric that leverages the concept of Word Mover's Distance to evaluate the quality of generated text. By considering the semantic similarity and movement of words, MoverScore provides a robust and context-aware evaluation metric. Its ability to capture semantic coherence and the fine-grained analysis of word correspondences makes it a valuable tool in assessing the quality of generated text in various natural language processing applications [23].

# CHAPTER 4

# EXPERIMENTAL RESULTS

In this case, we employed the Bert Model, developed by HuggingFace [9]. The Bert Model utilized for our purposes possesses certain characteristics: a vocabulary size of 30,522 words, a hidden size of 768, 12 hidden layers, 12 attention layers, and feed-forward layers consisting of 3,072 units. These specifications outline the essential framework of the transformers used in our approach.

Furthermore, an alternative version of the GPT-2 models, known as GPT-2 Medium, was introduced by OpenAI [24] in 2019. With a substantial parameter count of 355 million, GPT-2 Medium surpasses the smaller GPT-2 models in both size and computational power, although it remains smaller than the largest GPT-2 model, which boasts a staggering 1.5 billion parameters.

The primary objective of this model is to forecast the subsequent word in a sequence based on the preceding words. To achieve this, pre-training is carried out utilizing a causal language modeling objective, using an extensive corpus of English literature. Through this pre-training process, the model gains proficiency in generating coherent and contextually appropriate text by learning the underlying patterns and structures within the provided data.

We worked with the BCC New Datasets [25] and the Dailymail/CNN datasets [26].

## 4.1. DAILYMAIL/CNN DATASET

The statistic was sourced from the Dailymail/CNN news outlet. The collection includes 500 unique articles for various news stories. Extractive text summaries have been removed after going over every news item. The summary evaluation measure is computed when an extractive synthesis and a human-generated summary are compared.

TABLE I

| Rouge Score | Rouge 1 Recall | Rouge 1 Precision | Rouge 1 F-Measure | Rouge 2 Recall | Rouge 2 Precision | Rouge 2 F-Measure | Rouge L Recall | Rouge L Precision | Rouge L F-Measure |
|---|---|---|---|---|---|---|---|---|---|
| BERT | 0.47 | 0.23 | 0.29 | 0.18 | 0.08 | 0.1 | 0.44 | 0.21 | 0.27 |
| GPT 2 | 0.47 | 0.23 | 0.3 | 0.18 | 0.08 | 0.11 | 0.44 | 0.21 | 0.27 |
| KL SUMMARIZER | 0.32 | 0.24 | 0.27 | 0.12 | 0.08 | 0.09 | 0.3 | 0.22 | 0.25 |
| LUHN | 0.44 | 0.24 | 0.3 | 0.17 | 0.08 | 0.11 | 0.4 | 0.22 | 0.28 |
| LEX | 0.42 | 0.25 | 0.31 | 0.15 | 0.08 | 0.1 | 0.38 | 0.23 | 0.28 |
| WORD FREQUENCY | 0.49 | 0.21 | 0.29 | 0.2 | 0.07 | 0.1 | 0.46 | 0.2 | 0.27 |

The table provided displays the rouge scores for seven distinct models, namely Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

TABLE II

| BERT Score | Precision | Recall | F -Measure |
|---|---|---|---|
| BERT | 0.76 | 0.83 | 0.79 |
| GPT 2 | 0.76 | 0.83 | 0.8 |
| KL SUMMARIZER | 0.77 | 0.79 | 0.78 |
| LUHN | 0.76 | 0.82 | 0.79 |
| LEX | 0.77 | 0.81 | 0.79 |
| WORD FREQUENCY | 0.75 | 0.82 | 0.79 |

The table above presents the BERT scores for seven different models, which include Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

TABLE III

| MoverScore | Score |
|---|---|
| BERT | 0.56 |
| GPT 2 | 0.56 |
| KL SUMMARIZER | 0.55 |
| LUHN | 0.55 |
| LEX | 0.56 |
| WORD FREQUENCY | 0.55 |

The table provided displays the Mover scores for seven distinct models, namely Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

## 4.2. BBC NEWS DATASETS

The BBC News Databases encompass news archives from diverse sectors such as business, entertainment, politics, sports, and technology. Among the merged dataset of 2225 articles, there are 510 articles related to business, 386 articles on entertainment, 417 articles covering politics, 511 articles centered around sports, and 401 articles focusing on technology. For these 2225 articles, the table presented below illustrates the average approximate scores for recall, accuracy, and F-Measure in terms of the Rouge metric. Table V provides the BERT Score, while Table VI showcases the MoverScore.

TABLE IV

| Rouge Score | Rouge 1 | Rouge 1 | Rouge 1 | Rouge 2 | Rouge 2 | Rouge 2 | Rouge L | Rouge L | Rouge L |
|---|---|---|---|---|---|---|---|---|---|
| | Recall | Precision | F-Measure | Recall | Precision | F-Measure | Recall | Precision | F-Measure |
| BERT | 0.43 | 0.57 | 0.47 | 0.3 | 0.43 | 0.34 | 0.42 | 0.55 | 0.46 |
| GPT 2 | 0.43 | 0.58 | 0.48 | 0.3 | 0.45 | 0.34 | 0.42 | 0.57 | 0.47 |
| KL SUMMARIZER | 0.48 | 0.46 | 0.46 | 0.34 | 0.33 | 0.33 | 0.47 | 0.45 | 0.45 |
| LUHN | 0.75 | 0.56 | 0.63 | 0.65 | 0.46 | 0.53 | 0.74 | 0.55 | 0.63 |
| LEX | 0.66 | 0.53 | 0.58 | 0.53 | 0.43 | 0.47 | 0.64 | 0.52 | 0.57 |
| WORD FREQUENCY | 0.56 | 0.51 | 0.51 | 0.41 | 0.39 | 0.38 | 0.55 | 0.5 | 0.5 |

The table provided displays the rouge scores for seven distinct models, namely Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

Table V

| BERT Score | Precision | Recall | F Measure |
|---|---|---|---|
| BERT | 0.85 | 0.83 | 0.84 |
| GPT 2 | 0.86 | 0.83 | 0.84 |
| KL SUMMARIZER | 0.81 | 0.81 | 0.81 |
| LUHN | 0.83 | 0.88 | 0.85 |
| LEX | 0.83 | 0.86 | 0.84 |
| WORD FREQUENCY | 0.84 | 0.85 | 0.85 |

The table above presents the BERT scores for seven different models, which include Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

Table VI

| MoverScore | Score |
|---|---|
| BERT | 0.6 |
| GPT 2 | 0.6 |
| KL SUMMARIZER | 0.6 |
| LUHN | 0.64 |
| LEX | 0.63 |
| WORD FREQUENCY | 0.61 |

The table above presents the MoverScores for seven different models, which include Word Frequency, Lex, Luhn, Kl Summarizer, GPT-2 and BERT.

# CHAPTER 5

# CONCLUSIONS

Extractive summarization has emerged as a valuable tool for enhancing operational efficiency when confronted with copious amounts of material. By condensing lengthy texts into concise summaries, this approach enables users to navigate through information swiftly and effectively. However, it is important to acknowledge that extractive summarization does have limitations in terms of its ability to fully and accurately preserve the context and content of the original document.

Despite these shortcomings, extractive summarization finds utility across a wide range of activities, including the composition of news articles, academic papers, and legal documents. While the summarization process may not capture every intricate detail, it still offers significant benefits by distilling the main ideas and key points from extensive textual sources. By providing a condensed version of the content, extractive summarization empowers individuals to grasp the essence of a document quickly and efficiently.

In order to assess the effectiveness of extractive summarization techniques, we employed two prominent datasets: the Dailymail/CNN Dataset and the BBC News Dataset. These datasets serve as valuable resources for training and evaluating summarization models. By utilizing a diverse range of textual materials from reputable news sources, we aimed to ensure the robustness and generalizability of the summarization models.

To evaluate the performance of the summarization models, we employed several metrics such as Rouge, BERT, and MoverScore. These metrics allow for a comprehensive analysis of the summarization quality by comparing the generated model summaries with human-generated summaries. By assessing various aspects, including overlap in content and linguistic features, these metrics provide valuable insights into the efficacy of the extractive summarization process.

Based on the operating mechanism and performance evaluation, the top model for the Dailymail/CNN Dataset was found to be GPT-2. GPT-2 is a state-of-the-art language model

that has demonstrated exceptional capabilities in generating coherent and informative summaries. Its ability to capture important details while maintaining readability and coherence made it the preferred choice for the Dailymail/CNN Dataset.

On the other hand, for the BBC News Dataset, the best-performing model was identified as Luhn. Luhn's approach to summarization relies on the identification of key phrases in the text and their subsequent selection to form a coherent summary. This model demonstrated a remarkable ability to extract relevant information and construct summaries that effectively conveyed the essence of the original documents from the BBC News Dataset.

In conclusion, extractive summarization serves as a powerful tool for operating efficiently in the face of vast amounts of information. While it may not capture every aspect of the original context and content, it offers substantial benefits in various domains, including news articles, academic papers, and legal documents. The use of datasets such as the Dailymail/CNN Dataset and the BBC News Dataset, coupled with performance evaluation metrics like Rouge, BERT, and MoverScore, enables us to assess the effectiveness of summarization models. By identifying top-performing models like GPT-2 for the Dailymail/CNN Dataset and Luhn for the BBC News Dataset, we can leverage the strengths of these models to enhance the summarization process and improve operational efficiency.

# REFERENCES


1. https://www.turing.com/kb/how-bert-nlp-optimization-model-works
2. https://jalammar.github.io/illustrated-gpt2/
3. https://iq.opengenus.org/luhns-heuristic-method-for-text-summarization/
4. https://iq.opengenus.org/lexrank-text-summarization/
5. https://github.com/Tiiiger/bert_score
6. https://github.com/AIPHES/emnlp19-moverscore
7. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
8. Liu, Yang. "Fine-tune BERT for extractive summarization." ArXiv preprint arXiv:1903.10318 (2019). arXiv:1810.04805 (2018).
9. https://github.com/huggingface/transformers
10. https://jalammar.github.io/illustrated-word2vec/
11. https://demo.allennlp.org/next-token-lm
12. https://jalammar.github.io/illustrated-transformer/
13. https://karpathy.github.io/2015/05/21/rnn-effectiveness/
14. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
15. Liu, Peter J., Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. "Generating wikipedia by summarizing long sequences." *arXiv preprint arXiv:1801.10198* (2018).
16. Al-Rfou, Rami, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. "Character-level language modeling with deeper self-attention." In *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, pp. 3159-3166. 2019.
17. https://iq.opengenus.org/k-l-sum-algorithm-for-text-summarization/
18. H. P. Luhn, "The Automatic Creation of Literature Abstracts," in IBM Journal of Research and Development, vol. 2, no. 2, pp. 159-165, Apr. 1958, doi: 10.1147/rd.22.0159.
19. https://medium.com/@ashins1997/text-summarization-4e0b2cf252f2

20. Fang, Changjian, Dejun Mu, Zhenghong Deng, and Zhiang Wu. "Word-sentence co-ranking for automatic extractive text summarization." *Expert Systems with Applications* 72 (2017): 189-195.
21. Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." In Text summarization branches out, pp. 74-81. 2004.
22. Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. "Bertscore: Evaluating text generation with bert." ArXiv preprint arXiv:1904.09675 (2019).
23. Zhao, Wei, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. "MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance." arXiv preprint arXiv:1909.02622 (2019).
24. https://en.wikipedia.org/wiki/OpenAI
25. https://www.kaggle.com/datasets/gowrishankarp/newspaper-textsummarization-cnn-dailymail
26. https://www.kaggle.com/datasets/pariza/bbc-news-summary
27. https://en.wikipedia.org/wiki/Natural_language_processing
28. https://huggingface.co/docs/transformers/model_doc/xlnet

PAPER NAME

**thesis body (2).pdf**

WORD COUNT

**13775 Words**

CHARACTER COUNT

**79867 Characters**

PAGE COUNT

**61 Pages**

FILE SIZE

**2.2MB**

SUBMISSION DATE

**May 30, 2023 11:01 AM GMT+5:30**

REPORT DATE

**May 30, 2023 11:02 AM GMT+5:30**

## 11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 9% Submitted Works database

## Excluded from Similarity Report

- Bibliographic material
- Cited material