MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY ABDELHAMID IBN BADIS - MOSTAGANEM

Faculty of Exact Sciences and Computer Science

Department of Mathematics and Computer Science

Sector: Computer science

END OF STUDIES PROJECT

FOR OBTAINING THE MASTER'S DEGREE IN COMPUTER SCIENCE

Option: **Information Systems Engineering**

PRESENTED BY:

HABIBI LEILA

ELMECHERFI OMAR

THEME :

Natural Language Processing

Supported on:

In front of the jury composed of:

Last name and first name	Grade	Université de Mostaganem	President
Last name and first name	Grade	Université de Mostaganem	Examiner
Last name and first name	Grade	Université de Mostaganem	supervisor

Academic Year 2020-2021

Dedication

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, whose words of encouragement and push for tenacity ring in my ears, my sister Amira and my Brother Abderahmane Oussama.

I also dedicate this dissertation to my friends Linda and Oumrania, and to all others members family without exception who have supported me throughout the process. I will always appreciate all they have done.

Last dedicate go to my partner and my best friend ELMECHERFI Omar.

I am grateful to you all and promise not to fail you.

HABIBI Leila

Dedication

I would like to dedicate this dissertation to my parents, my brothers NADIR and ABDERAHMANE, my sister IKRAM and all family members who never ceased to give their encouragements, love and support in every way. No less gratitude goes to my friends HAMZA and NOUREDDINE, and my dear best friend LEILA HABIBI whom without her interest and co-operation I could not have produced this study.

I am grateful to you all and promise not to fail you.

ELMECHERFI Omar

Acknowledgments

Our first and most earned, expression of gratitude and thankfulness go to our supervisor and advisor, Mr MOUMENE Mohammed EL Amine for his consistent availability, willingness to listen, guidance and support, whose useful discussions and scholarly notes have developed and reshaped our reflection and understanding, and above all, for being the ideal source of inspiration and motivation.

No less gratitude to our co-supervisor Mr MECHAOUI Moulay Driss for his time, guiding and help during preparing our thesis.

We also would like to express our gratitude again and thank a cousin TABET Shaymae for helping us to develop our English language skills.

We thank the board of examiners for having accepted to read our work and be part of the committee.

We will be forever grateful.

Abstract

The increasing availability of online information has triggered an intensive research in the area of automatic text summarization within the Natural Language Processing (NLP). The various methods to summarize one or more documents can be broadly classified into extractive and abstractive text summarization where the former involves selecting key parts in the document and embedding into the summary while balancing between salience and redundancy which this work mainly focused on. in this thesis a comparative analysis is proposed for five algorithms: LexRank, KLSum, Luhn, LSA and BERT. The experiments on the MultiLing 2013 show the best algorithms among the five proposed based on the parameters: F-Score, precision and recall in ROUGE-1, ROUGE-2 and ROUGE-L. After comparing the results obtained, we noticed that the BERT algorithm gives the best results in terms of precision and F-Score on all the metrics tested. On the other hand, in terms of recall, the LSA algorithm is considered to be better.

Keywords

Summary, Automatic Summarization, Extractive summary, Natural Language Processing, abstractive summary, ROUGE metric, evaluation summary.

List of Figures

Tables list

Abreviations list

Abreviation	Full expression	Page
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers	21
RNN	**R**ecurrent **N**eural **N**etwork	22
NLP	**N**atural **L**anguage **P**rocessing	10-21
MLM	**M**asked **L**anguage **M**odel	23
NSP	**N**ext **S**entence **P**rediction	24
CLS	**C**ross-**L**ingual **S**ummarization	25
MHAtt	**M**ulti-**H**ead **Att**ention	27
LSTM	**L**ong **S**hort-**T**erm **M**emory	27
K-L sum	**K**ullback-**L**ieber Sum	28
LSA	**L**atent **S**emantic **A**nalysis	21-32
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency	18
SVD	**S**ingular **V**alue **D**ecomposition	34
NLTK	**N**atural **L**anguage **T**ool**K**it	40
ROUGE	**R**ecall **O**riented **U**nderstudy for **G**isting **E**valuation	13
AI	**A**rtificial **i**ntelligence	6
RST	**R**hetorical **S**tructure **T**heory	12
RS	**R**hetorical **S**tructure	12
POS	**P**art of **S**peech **T**agging	17
NER	**N**ame **E**ntity **R**ecognition	17
BoW	**B**ag **O**f **W**ord	26

Table of Contents

General Introduction

Automatic Natural Language Processing (NLP) is a growing transdisciplinary field that has led to various fields of application: automatic error correction, text analysis, automatic generation of summaries, translation assistance, sentiment analysis and so on. In addition, the need for NLP applications is becoming more and more essential with the mass of information available, of which human language remains a predominant source of information.

Along with the growth of the internet and big data, making people overwhelmed by the large information and documents on the internet. This triggers the desire of many researchers to develop a scientific approach that can summarize texts automatically. Moreover, automatic text summarization generates summaries containing important sentences and includes all-important relevant information from the original document. So, the information quickly arrives and does not lose the original intent of the document. The area of text summarization research has been studied since the mid-20th century. Many different approaches have been created to date. Based on the number of the document, there is single and multi-document summarization. Meanwhile, based on the summary results there are the extractive and abstractive results. However, extractive summarization is a summary that consist entirely of extracted content so that the results of summary are sentences or words obtained from the original text.

In this research, automatic text summarization will be tackled and studied. while extractive approach and its algorithms will be highlighted as a major point in this dissertation.

In this light, this dissertation is dedicated to answer the following question:

- Which extractive algorithm is able to provide us with a perfect summary of a given document or an article?

As a possible answer to the aforementioned question, the following hypothesise is suggested:

- There are five extractive algorithms that are going to be tested and studied in details in our research. However, BERT and LSA might be most suitable algorithms for a good summary.

To this end, this work has been conducted and divided into four chapters. The first one, titled introduction to text summary, is devoted to main definitions about the given topic. And it sheds the light on the main steps of text summarization and its evaluation methods. It is also devoted to explain types and methods of automatic text summarization. While the second chapter, titled state of the art on extractive methods, sheds the light on extractive method and its algorithms as the most important point in this research. Furthermore, extractive methods are detailed and well explained in this chapter. While in the third chapter a comparative study of extractive methods is provided. The chapter is divided into two main parts. The first part deals with methods' implementation. While, the second part of the chapter is devoted to the ROUGE metric and its implementation. Finally, the comparison's results are presented in the fourth and last chapter.

Chapter 1

Introduction to Automatic Text Summarization

1.1 Introduction

Within the growth in information, people are nowadays obliged to read plenty of pages and articles just to find one particular information needed. And this is considered as time consuming. Moreover, it has become very important to provide upgraded mechanisms to extract useful information. Mechanisms such as Automatic Text Summarization system. It Enables people to extract brief information from large volumes of texts available in documents. In other words, a text Summarizing System is an important tool that helps in finding a useful piece of information from a vast amount of data.

In the studies of Radev et al. [1'] and Das and Martins [2'], the aspects of a summary are defined as follows: first, a summary can be created using single or multiple documents. Second, a summary contains all necessary information and it does not include redundant information. Third, a summary is short, at least shorter than half of the original document [1].

In this chapter, text summarization will be defined. Then we will shed the light on the early history of text summarization. Moreover, the categorization of text summarization systems will be discussed. After that, the approaches of text summarization will be presented. Then we go through the main steps of text summarization. Finally, the chapter will be concluded with an evaluation of summarization systems.

1.2 Definition

Text summarization is the process of creating a short and coherent version of a longer document. More simply. It is the procedure of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks).

Automatic text summarization can be defined also as a technique that identifies the most important information from the whole document to present the original text in a shorter version

while keeping its main content. Which means omitting irrelevant information and minimizing details to generate a compact coherent summary document. The work on solving this problem falls into the field of natural language processing (NLP).

1.3 The early history of text summarization

Interest in automatic text summarization, arose as early as the fifties. Most of the early work on text summarization focused on single document summarization in scientific articles. Due to the lack of powerful computers and technological developments, summarization systems considered some simple surface level features of sentences like word frequency, position, length of the sentence and so on.

In 1970's Artificial Intelligence (AI), technology was developed and most of the summarization systems relied on AI technology. In 1980, several summarization systems are developed based on cognitive science theory. In 1990, Information retrieval methods are used for domain independent summarization. In 1995, machine-learning techniques are developed and it is highly used in summarization systems. Now the statistical and mathematical techniques are widely used for extractive text summarization [6].

The technological developments and its advantages and disadvantages are explained in Table 1**.**

Year	Methods	Advantages	Disadvantages
1958	Simple surface level features of sentences.	The sentences, which include most frequent words, are selected as summary sentences.	Duplication in summary sentences.
1970	Artificial Intelligence.	Frames or templates are used to identify the conceptual relation of entities and extract the relation between entities by an assumption.	Only limited frames or templates may lead to incomplete analysis of conceptual entities.

1980	Cognitive science theories	The system can overcome the redundancy in some extent. Extract the representative sentences from the source text.	Complex task and limited to specified area.
1990	Information retrieval techniques	Generate significant sentence from source text same as information retrieval techniques.	Doesn't consider the semantic aspects such as synonymy and polysemy
1995	Machine learning techniques	Different machine learning algorithms are used and provides more generalized summary.	Computationally complex and lack of semantic analysis of source text.
1997	Statistical and Algebraic methods	Depended on some heuristics, linguistics and mathematical techniques. Easy to implement.	Without any syntactic analysis of the source text.

Table 1 : The history of text summarization

1.4 Categorization of text summarization systems

Text summarization could be categorized based on different criteria, which means that summaries can have different forms. Extractive or abstractive according to the way the summary is created. Also, they can be categorized by the number of input documents used: single document or multi document summarization. Summarization system can also be categorized as generic and query-based and so on. However, the most comprehensive criteria for classifying summaries are covered in this chapter. These categories, which are summarized in figure 1, are as following [3]:

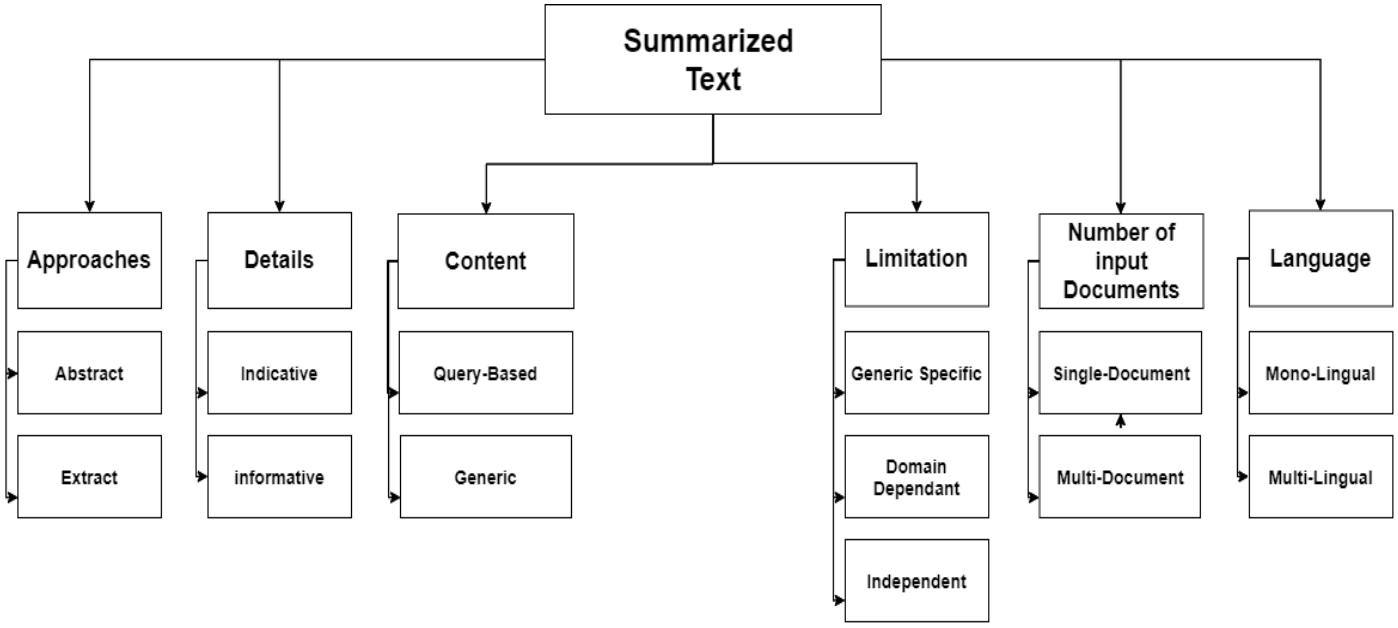


Figure 1: Categorization of text summarization systems

1.4.1 Approaches

There are two main schools of thought for how the summary is generated: extractive and abstractive summarization. Extractive summary method extracts the most important and meaningful sentences or phrases having highest scores from the original text and placing them together to a new shorter text without any changes. Abstractive summary method builds new sentences which may not be present in the original text after analyzing and understanding the original, that why abstractive summarization approaches is similar to the way that human summaries.

1.4.2 Details

Based on the summarization purpose, type of details or style of output, the summarization can be Indicative or Informative. Indicative summary presents the main idea of the entire text to the user; it gives a quick view of the original text. The typical length of this type of summarization is 5 to 10 percent of the main text. On the other hand, Informative summary give concise information of the original text to the user. The length of informative summary is 20 to 30 percent of the main text [3].

1.4.3 Content

Based on the content type of the original text, the summarization may be considered as Generic or Query based. In Generic summarization, extracted information is not a user specific and does not depend on the subject of the document. In this system, the user does not have any previous understanding of the text and it addresses broad community of readers, so all the information is in the same level of importance. While in Query-based summarization, the generated summary based on the user's query. In other words, the user has to determine the topic of original text in a form of a query, before the summarization process. In these types, the user has general information about the text and searches for specific information, which is usually an answer to a question. So, the user asks that special information in form of a query and the system only extract that information from the text and presents it as a summary [3].

1.4.4 Limitation

Another classification is based on limitations on input text. Three different summaries can be defined, genre specific, independent and domain dependent. Genre specific systems accept only special type of input text such as, stories, newspaper articles and so on. The system uses the structure of these templates for generating the summary. On the other side, independent systems can accept different types of texts, as they are not relied on the domain. Moreover, Dependent systems summarize the texts, which their subject defined in the fixed domain.

1.4.5 Numbers of input documents

A summarization system can be one or more documents as input. It can be divided into Single Document or Multi-Document Summarization. Single document accepts only one document as input while multi document must have more than one document, in this case, documents must have topic relation with each other and the summarizer generates a summary based on them.

1.4.6 Language

Based on the language of the input text and the generated summary, summarization systems can be classified in two main groups, monolingual and multi lingual. Mono Lingual systems deal with documents with a specific language, like English, and the generated summary

is based on that language too. On the contrary, Multi-Lingual System can accept documents in different languages and generated summaries are in these different languages. The user can choose the language of the output summary.

1.5 Text Summarization Approaches

In order to generate a high-quality summary, different NLP techniques must be used like language analysis, information inference and so on. Moreover, there are three different approaches for scoring and selecting sentences, which are discussed below and presented in the Figure 2.

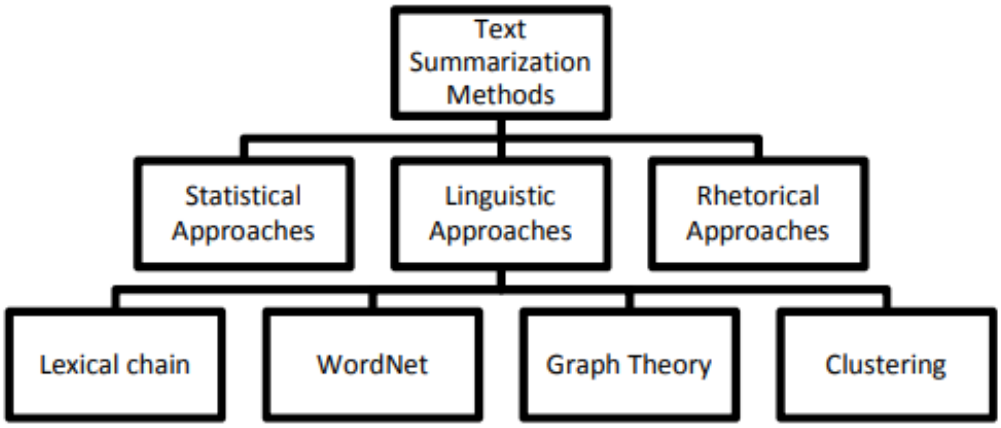


Figure 2 : Text Summarization Approaches [3]

1.5.1 Statistic approaches

In Statistical methods, sentence selection is done based on word frequency, indicator phrases and other features regardless of the meaning of the words. These methods are based on the idea that text surface cues are the most obvious indication of the text contents [3].

1.5.2 Linguistic approaches

Linguistic approaches are based on considering the connections between words and trying to find the main concept by analyzing the words. There are some methods such as, Lexical chain, WordNet, Graph theory, clustering and so on.

1.5.2.1 Lexical Chain

Lexical chain produces a presentation of text contiguous structures. Basically, lexical chains exploit the cohesion among an arbitrary number of related words. Lexical chains can be computed in a source document by grouping (chaining) sets of words, which are semantically related. Identities, synonyms, and hyponyms are the possible relations among words that might be used to group them into a same lexical chain [3]. A lexical chain is used for information retrieval and grammatical error corrections.

1.5.2.2 WordNet

WordNet is a lexical database of semantic relations between words in more than 200 languages, specifically designed for NLP. It links words into semantic relations including synonyms, hyponyms, meronyms and so on. then it can be seen as a combination and extension of a dictionary and thesaurus.

1.5.2.3 Graph theory

Graph methods, represent the documents as a connected graph. Sentences form the vertices of the graph and edges between the sentences indicate how similar the two sentences are, which means that node are the sentences. Two sentences are connected with an edge if they share common worlds**,** this binding relationship called the measure of similarity. This representation gives two results: First, the partitions (sub-graphs) included in the graph, create discrete topics covered in the documents. The second is the identification of the important sentences in the document. Sentences that are connected to many other sentences in the partition are possibly the center of the graph and more likely to be included in the summary.

1.5.2.4 Clustering

Clustering is used to decrease the information by categorizing and grouping similar data. There are two major types of clustering; hierarchy clustering and partitioning. In hierarchy, clustering smaller clusters are mixed with each other to form bigger clusters or sometimes the clusters are divided by half. On the other hand, partitioning tries to decompose the data collection to distinct clusters [3].

1.5.3 Rhetorical approaches

Rhetorical structure theory (RST) is based on the Rhetorical connections between different parts of the text. In this theory, the Rhetoric behind the decomposed text is extracted. In summarization systems, a Rhetorical Structure (RS) represents the logical connections between different parts of the text and interprets these connections. This information represents the discourse structure and features of the main document. After identifying text units and rhetorical connections between them, the RS tree is formed based on this information [3].

1.6 Main steps for text summarization

There are three main steps for summarizing documents. Topic identification, interpretation and summary generation.

1.6.1 Topic identification

In this step, the most prominent information in the text is identified. There are different techniques for topic identification such as position, Cue Phrases and word frequency. Methods that are based on the position of phrases are the most useful methods for topic identification. The orderly structure of texts results to extract the main information based on their position.

1.6.2 Interpretation

Abstract summaries need to go through an interpretation step. In this step, different subjects are fused in order to form a general content.

1.6.3 Summary Generation

In this step, the system uses text generation method which itself is still an open research topic. This step includes a range of various generation methods from very simple word or phrase printing to more sophisticated phrase merging and sentence generation. In text generation, the computer generates the natural language from the processed information of the previous steps [3].

1.7 Evaluating the summarization systems

Summary evaluation is a very important aspect for text summarization. Evaluation methods are used to evaluate the usefulness and trustfulness of the summary. In summary, evaluating the qualities like comprehensibility, coherence, and readability is difficult. Experts who compare different summaries and choose the best one might perform the system's evaluation manually. A problem with this approach is that the individuals who perform the evaluation task normally have very different ideas on what a good summary should contain. Generally, summaries can be evaluated using intrinsic or extrinsic measures. While intrinsic methods attempt to measure summary quality using human evaluation and extrinsic methods measure the same through a task-based performance measure such the information retrieval-oriented task.

1.7.1 Human Evaluation

The simplest way to evaluate a summary is to have a human assess its quality; this human must be an expert in this domain to have a better evaluation of the system summarized.

1.7.2 Automatic Evaluation Methods

There has been a set of metrics to automatically evaluate summaries. ROUGE is the most widely used metric for automatic evaluation [5].

ROUGE tool is used to evaluate text summarization, which consist of precision, recall and F-score.

1.8 Conclusion

Due to the exponential growth of the internet, the amount of information available has become very important. Therefore, it is difficult for humans to summarize large amounts of text.

Thus, there is an immense need for automatic summarization tools in this age of information overload. Currently, text summarization is one of the active areas of research and attracts many researchers from different fields. Automated summarization is an important area in research. It consists of automatically creating a summary of one or more texts. It can also be categorized in to various groups based on different approaches that were presented in this dissertation. As discussed earlier, there are three main steps for producing a summary from an input text. Topic identification, interpretation and summary generation. Most of summarization systems follow these steps in order to generate a summary.

Chapter 2

State of the art on extractive methods

2.1 Introduction

The text summarization is usually classified as abstractive and extractive summarization. Abstractive summarization is a technique in which the system understands the original text and then creates the summary for the document. First, it understands the main concepts of a document then it expresses those concepts in fewer words. On the other hand, Extractive summarization technique involves the selection of important sentences, paragraphs, etc. From the original document(s) and then combines them to create a summary. The importance of sentences is decided based on statistical and linguistic features of sentences.

This chapter focuses on the extractive text summarization. It is organized as follows starting with the general flow of extractive summarization as a main point. Extractive summarization features will be discussed in details as a second point. After that, lights will be put on the methods of extractive summarization. Then, text extractive summarization algorithms will be explained. Finally, problems with extractive summary will be mentioned.

2.2 The general flow of extractive summarization

An extractive based method identifies key phrases from the input documents to generate summary. Original sentences are not modified in extractive based method. The Main phases are presented in the Figure 3.

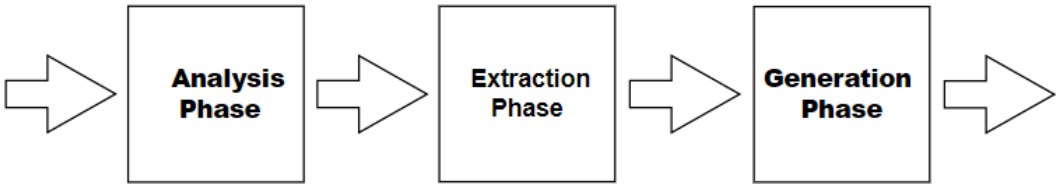


Figure 3 : Architecture of extractive based methods

Input: Text

- **Analysis phase**

  Pre-processing

  Pre-processing is a structured representation of the original text. It usually includes:

  - **Sentence segmentation:** To manipulate text at the word level, the prerequisite is to divide the text into individual sentences. Sentence segmentation converts raw text into sentences, as a list of strings
  - **Tokenization:** tokenization is a dividing the text to smaller units, which are often words. Tokenization takes a list of sentences as an input and provides a list of tokens as output.
  - **Stemming:** this stage roots extraction of words without considering different scenarios such as singular or plural, the time and all prefix and suffix.
  - **Part of speech tagging (POS Tagging):** POS tagger analyses output of Tokenization or sequence of words, and assigns appropriate part of speech tag to each word. POS is useful in extraction of nouns, adverbs, adjective, which provide some meaningful information about text. POS tagging takes a list of tokens as input and generates a list of tuples with POS annotation.
  - **Entity detection:** Entity detection provides the identification of predefined categories such as person, location, quantities, percentage, organizations, expressions of times, monetary values and so on [13].

The system called Name Entity Recognition (NER) provides the entity detection linguistic processing. The NER system uses linguistic grammar-based techniques and also statistical models to identify the entity. Entity detection takes POS annotation tuples and provides the output in form of a tree. Here, the input is the POS annotation tuples and provides the output with NER values such as person, organization and so on.

- **Relation detection:** Relation detection identifies the possible relation between two or more chunked sentences. One of the features, co-reference words, provides the link between pronouns and its corresponding nouns. Co-reference chain provides a relation between two or more sentences. Co-reference is useful in replacement of the pronouns with proper nouns. Relation detection step takes a list of trees of parsed string as input and provides the list of tuples as output.

- **Extraction phase**

  Selection of sentences ranking method can be used for extraction phase. Hence, higher ranked sentences are preferred for summary.

- **Generation phase**

  Re-ranking can be done using similarity measures to minimize redundancy.

2.3 Extractive summarization features

Extractive text summarization features find appropriate sentences, select important sentences from the original document, and include these sentences to the summary. The selection of these important sentences involves assigning a score to sentences based on a countenance that are predefined based on the methodology applied. Both word level and sentence level features are employed in text summarization literature.

2.3.1 Word level features

- **Keyword features:** Keywords are essential in identifying the importance of the sentence. The sentence that consists of main keywords is most likely included in the final summary. The keywords can be verbs, nouns, adjectives or adverbs and are determined based on the TF-IDF method.
- **Title Word feature:** The sentences consist of words in title of the source document have more chances to be included in final summary as it depicts the theme of the document.
- **Cue phrase feature:** The sentences containing cue words (in conclusion, because, furthermore, therefore) are more likely to be in the final summary. Cue phrases are the words that would affect positively or negatively to the respective sentence weight to indicate significance of the summary [7].
- **Biased word feature:** The sentences that consist of biased words are more likely important. The biased words are a list of the predefined set of words that may be domain specific. They are relatively important words that describe the theme of the document [7].
- **Term frequency:** The most occurring words increases the score of the sentences. TF-IDF is used to calculate the frequency of each word and the importance of the sentence which increases for a greater number of times the word is visible in the sentence. The sentences containing main keywords are more likely to occur in final summary.
- **Upper case word feature:** The words that contain only upper case or called abbreviation can be reflect on the summary and sentences holding these abbreviations can be included in the summary [8].

2.3.2 Sentence level features

- **Sentence location feature:** The sentences that occur in the beginning and the conclusion part of the document are most likely to be in the final summary. It is because of the intuition that most of the document have hierarchical structure with important information in the starting and the ending of the paragraph.

- **Sentence length feature:** The length of the sentences is an essential feature in selecting sentences to be included in the summary and which not to be. Neither the too short sentence nor the too long sentence is suitable for the summary. The normalized length of the sentence is calculated as the ratio between the number of words in the sentence and the number of words in the longest sentence in the document.
- **Paragraph Location:** Similar to the sentence location, the location of the paragraph is also important for the selection of the important sentence.
- **Similarity or cohesion feature:** Similarity feature is necessary to remove redundancies and reorder the segments to obtain a coherent summary. Similarity can be calculated among the sentences [8].

2.4 Extractive text summarization methods

Extractive text summarization methods can be classified to supervise learning and unsupervised learning methods.

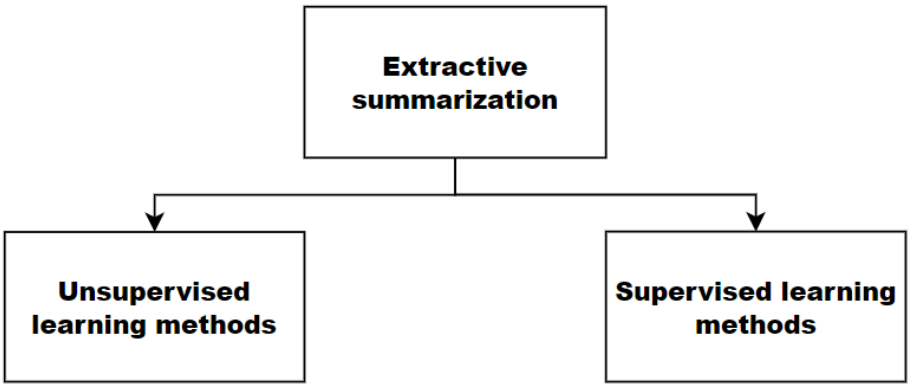


Figure 4 : Overview of Extractive Summarization methods

2.4.1 Supervised Learning Method

The supervised learning techniques are based on a classification approach at the sentence level where the system learns using the examples to classify between summary and non-summary sentences.

2.4.2 Unsupervised Learning Method

Unsupervised learning techniques do not require any human input for deciding the valuable features of the document. Moreover, unsupervised summaries provide a higher level of automation compared to supervised models and are more suitable for processing Big Data [7].

Categories	Methodology	Concept	Advantages	Limitations
SUPERVISED LEARNING APPROACHES	Machine Learning approach	Summarization task modelled as classification problem	Large set of training data improves the sentence for summary	Human interruption required for generating manual summaries
UNSUPERVISED LEARNING APPROACHES	Graph based Approach	Construction of graph to capture relationship problem	1- Captures redundant information 2-Improves coherency	Doesn't focus on issues such as dangling anaphora problem
UNSUPERVISED LEARNING APPROACHES	TF-IDF	Statistical measure that evaluates how relevant a word is to a document in a collection of documents	1-Easy to compute 2- have some basic metric to extract the most descriptive terms in a document	TF-IDF is based on the bag-of-words (BoW) model, therefore it does not capture position in text, semantics, co-occurrences in different documents, and so on .

UNSUPERVISED LEARNING APPROACHES	Query based method	The sentences in a given document are scored based on the frequency counts of terms (words or phrases)	The sentences containing the query phrases are given higher scores than the ones containing single query words.	Of this approach is that it also extracts some of the headings with the sentences
UNSUPERVISED LEARNING APPROACHES	LSA	Automatic statistical and mathematical technique. It is used for extracting and concluding relations of expected contextual usage of words in passages of discourse	can handle Synonymy problems to some extent.	1- The number of dimensions to be used must be same as to the number of the sentences needed for summary 2- Sentences with high index value will not be chosen although it is suitable for the summary.

Table 2 : SUPERVISED AND UNSUPERVISED LEARNING METHODS FOR TEXT SUMMARIZATION

2.5 Extractive text summarization algorithms

There is a large collection of algorithms for extractive text summarization approach. However, our study will be limited only by the few algorithms that are going to be presented in this dissertation.

2.5.1 BERT

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a new pre-trained model that is naturally bidirectional used to overcome the limitations of Recurrent neural network (RNN) and other neural networks as Long-term dependencies. This pre-trained model can be tuned to easily perform the NLP tasks as specified, Summarization in our case (BERT for text summarization)**.**

Bert is an automatic natural language processing (NLP) algorithm based on neural networks. Thus, it allows the analysis of words in relation to all the others in the sentence,

rather than word by word. It gives meaning to conversational sentences and that in all languages.

2.5.1.1 Architecture

The general architecture of BERT is shown in the Figure 5.

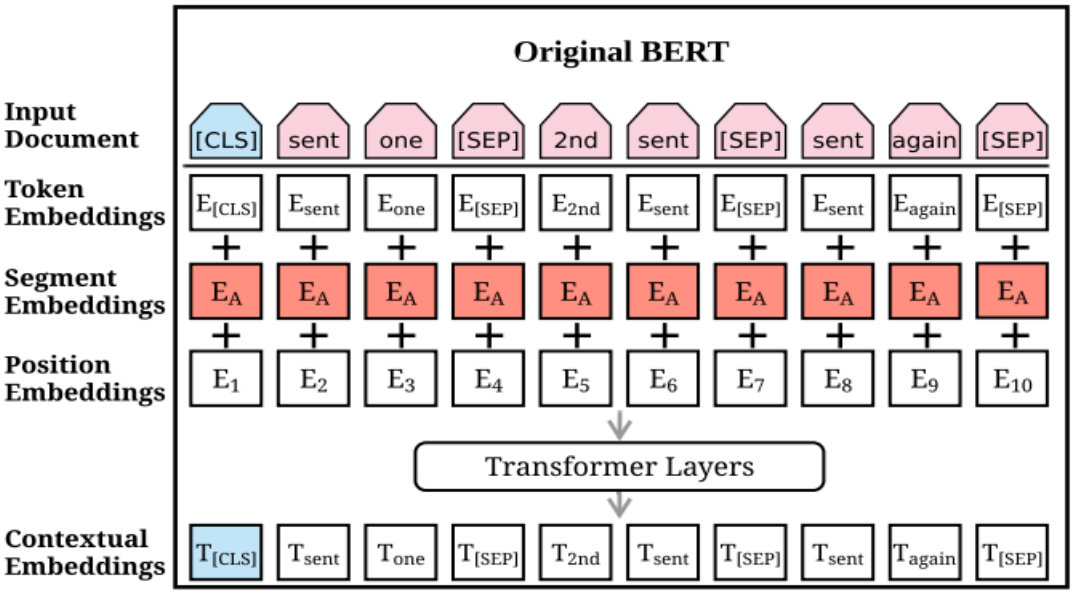


Figure 5: Architecture of the original BERT model

The sequence on top is the input document, followed by the summation of three kinds of embedding for each token. The summed vectors are used as input embedding to several bidirectional Transformer layers[1], generating contextual vectors for each token.

There are two BERT models introduced:

- BERT base: In the BERT base model we have 12 transformer layers along with 12 attention layers and 110 million parameters.

[1] Transformer layer is actually a combination of complete set of encoder and decoder layers and the intermediate connections. Each encoder includes Attention layers along with a RNN. Decoder also has the same architecture but it includes another attention layer in between them, as does the seq2seq model. It helps to concentrate on important words.

- BERT Large: In BERT large model, we have 24 transformer layers along with 16 attention layers and 340 million parameters.

2.5.1.2 Methodology

Let d denote a document containing several sentences [$sent_1$, $sent_2$, ⋯, $sent_m$], where $sent_i$ is the i-th sentence in the document. Extractive summarization can be defined as the task of assigning a label $y_i \in \{0,1\}$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

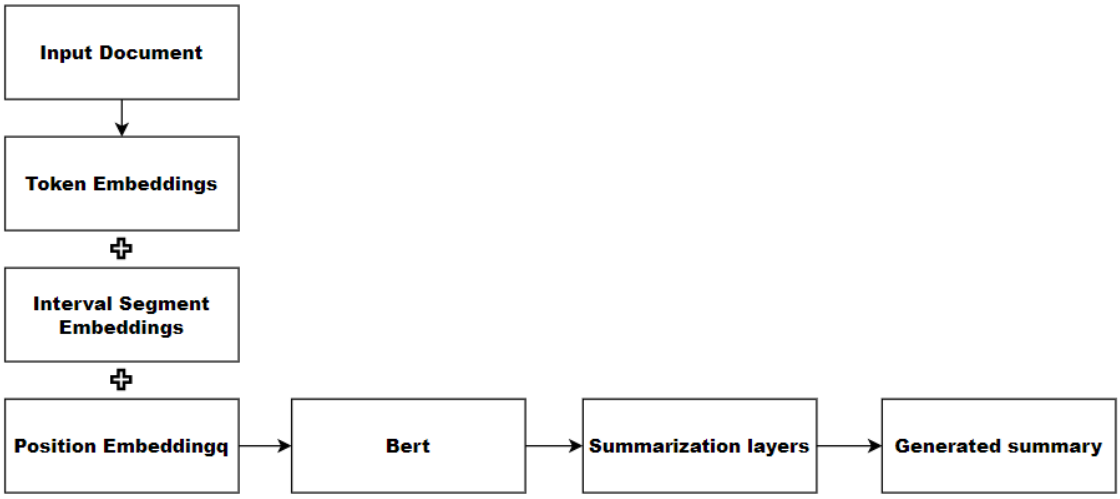


Figure 6: extractive text summarization BERT

There are two steps in this framework:

- **Pre-training:**

During pre-training, the model is trained on unlabeled data over different pre-training tasks.

Task 1: Masked Language Model (MLM):

Masked Language Modeling is a fill-in-the-blank task, where a model uses the context words surrounding a mask token to try to predict what the masked word should be.

For an input that contains one or more mask tokens, the model will generate the most likely substitution for each.

Task 2: Next Sentence Prediction (NSP):

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.

- **Fine-tuning:** the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

2.5.1.3 Extractive Summarization with BERT (BertSum):

To use BERT for extractive summarization, we require it to output the representation for each sentence. However, since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences. Meanwhile, although BERT has segmentation embedding for indicating different sentences, it only has two labels (sentence A or sentence B), instead of multiple sentences as in extractive summarization. Therefore, we modify the input sequence and embedding of BERT to make it possible for extracting summaries.

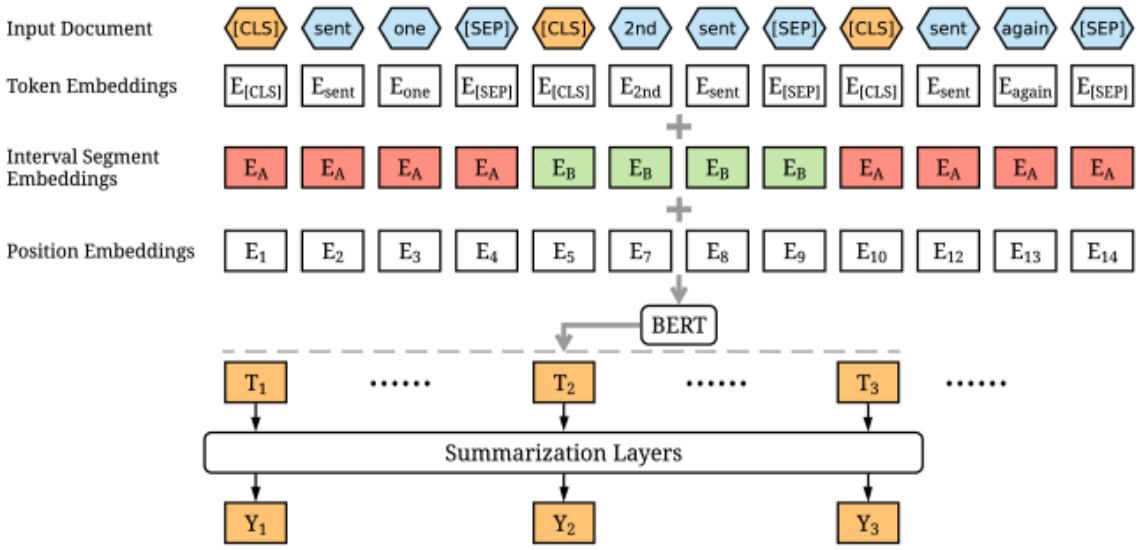


Figure 7: The overview architecture of the BERTSUM model

The complete process can be divided into several phases, as follows:

- **Encoding multiple sentences**

As illustrated in Figure 1, [CLS] token is inserted before each sentence and a [SEP] token after each sentence. The [CLS] is used as a symbol to aggregate features from one sentence or a pair of sentences. The model is modified by using multiple [CLS] symbols to get features for sentences ascending the symbol.

- **Interval segment embedding**[2]

We use interval segment embedding to distinguish multiple sentences within a document. For $sent_i$ we will assign a segment embedding EA or EB conditioned on i is odd or even. For example, for [sent1, sent2, sent3, sent4, sent5] is assigned [EA, EB, EA, EB, EA].

[2] Embedding: It basically refers to the representation of words in their vector forms. It helps to make their usage flexible. It helps in unlocking various functionalities for the semantics from understanding the intent of the document to developing a similarity model between the words.

The vector T_i that is the vector of the i-th [CLS] symbol from the top BERT layer will be used as the representation for senti.

- **The embedding types**

There are three types of embeddings applied to our text prior to feeding it to the BERT layer, namely:

- Token Embeddings - Words are converted into a fixed dimension vector. [CLS] and [SEP] is added at the beginning and end of sentences respectively.
- Segment Embeddings - It is used to distinguish or can be classified the different inputs using binary coding.
- Position Embeddings - BERT can support input sequences of 512. Thus, the resulting vector dimensions will be (512,768). Positional embedding is used because the position of a word in a sentence may alter the contextual meaning of the sentence and thus should not have same representation as vectors.

Notes

- ✓ Every word is stored as a 768-dimensional representation.
- ✓ The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.
- ✓ BERT uses a very different approach to handle the different contextual meanings of a word.

2.5.1.4 Summarization layers

The one major noticeable difference between RNN and BERT is the self-attention layer. The model tries to identify the strongest links between the words and thus helps in representing.

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

- **Simple Classifier:** In a simple classifier method, a linear layer is added to the BERT along with a sigmoid function to predict the score $\hat{Y}_i$.

$$\hat{Y}i = \sigma\ (WoTi + bo)$$

- **Inter Sentence Transformer:** In the inter sentence transformer, the simple classifier is not used. Rather various transformer layers are added into the model only on the sentence representation thus making it more efficient. This helps in recognizing the important points of the document.

$$h^{\sim L} = LN\ (h^{L-1} + MHAtt\ (h^{L-1})$$

$$h^{L} = LN(h^{\sim L} + FFN(h^{\sim L}))$$

where h^0 = PosEmb(T) and T are the sentence vectors output by BERT, PosEmb is the function of adding positional embeddings (indicating the position of each sentence) to T, LN is the layer normalization operation, MHAtt is the multi-head attention operation and the superscript l indicates the depth of the stacked layer.
These are followed by the sigmoid output layer:

$$\hat{Y}_i = \sigma\ (W_o h L_i + b_o)$$

h^L is the vector for $sent_i$ from the top layer (the L-th layer) of the Transformer.

- **Recurrent Neural Network:** Although the Transformer model achieved great results on several tasks, there are evidence that RNN have their advantages, especially when combining with techniques in Transformer. Thereore**,** An LSTM layer is added with the BERT model output in order to learn the summarization specific features. Where each LSTM cell is normalized. At time step *i*, the input to the LSTM layer is the BERT output T_i.

$$C_i = \sigma(F_i).C_{i-1} + \sigma(I_i).\tanh\ (G_{i-1})$$

$$h_i = \sigma(O_t).\tanh(LN_c(C_t))$$

where F_i, I_i, O_i are forget gates, input gates, output gates; G_i is the hidden vector and C_i is the memory vector, h_i is the output vector and LNh, LNx, LNc are there difference layer normalization operations. The output layer is again the sigmoïde layer.

2.5.2 KL-SUM

KL-sum algorithm (Kullback-Lieber (KL) Sum algorithm) for text summarization, which focuses on minimization of summary vocabulary by checking the divergence from the input vocabulary.

It is a sentence selection algorithm where a target length for the summary is fixed (L words). It is a method that greedily adds sentences to a summary so long as it decreases the KL Divergence.

2.5.2.1 Analysis

In mathematical statistics, the KL divergence (relative entropy) is a measure of how one probability distribution is different from another. Less the divergence, more the summary and the document are similar to each other in terms of understandability and meaning conveyed.

2.5.2.2 Mathematical analysis

K-L sum algorithm introduces a criterian for selecting a summary **S** from given document **D**,

$$S^{*}=\min(KL(P_D|| P_S))$$

$$S: [words(S) <= L$$

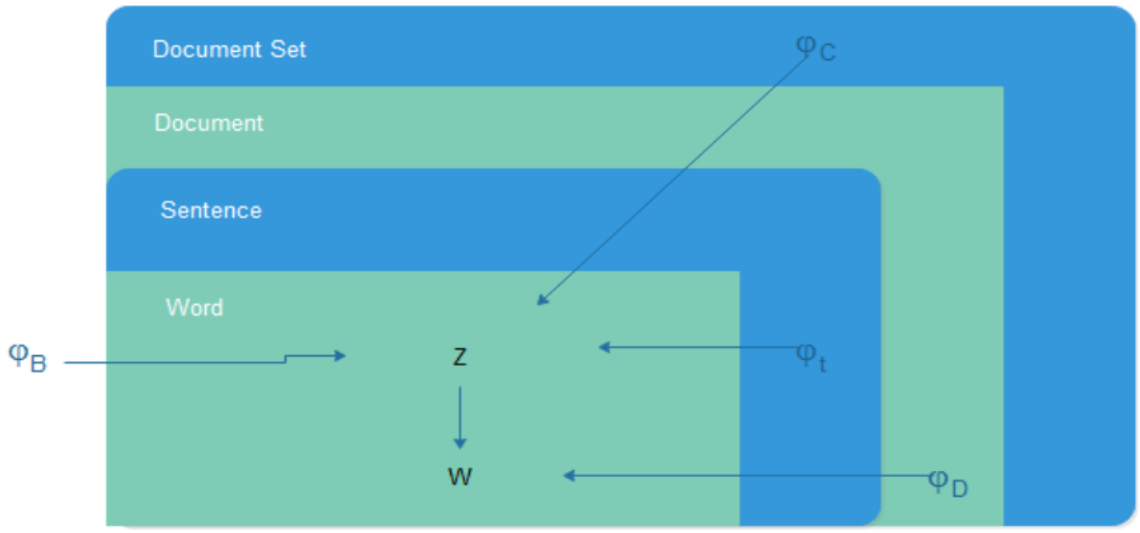


Figure 8: Document set architecture

Where P_s is the empirical unigram distribution of the candidate summary-S and KL (P|| Q) represents KL divergence given by:

$$\Sigma\ P(w)\log P(w)/Q(w)$$

This quantity represents the divergence between true distribution P (here document set unigram) and the approximating distribution Q (the summary S distribution). This summarization method finds a set of summary sentences, which closely match the document, set unigram distribution.

2.5.2.3 Algorithm

It uses greedy optimization approach:

1. Set S={} and d=0
2. while | S|<= L do:
3. for *i* in [1...N_D] , d_i=KL(P_s|| P_D)
4. Set S=S + S_i with minimum d_i and d=d_i
5. Stop if there is no *i* such that d_i:<*d*

The last issue is to set order of the selected sentences. They are ordered according to the value of P_i. The method adopted is to follow the order in the source documents, that is: for

each selected sentence S_i extracted from document D_j, compute a position index p_i (in [0..1]) which reflects the position of S_i within D_j. The sentences in the target document are ordered according to the value of p_i.

2.5.2.4 Computing KL Divergence

KL measures the divergence between a probability distribution P and Q. Think of P as (x1: p1, x2: p2... xn:pn) with $\Sigma pi = 1.0$ and Q as (y1:q1, ..., yn:qn) with $\Sigma qj = 1.0$. Assume for now that P and Q are defined over the same outcomes x_i - we will discuss the possibility for differences below: Then the definition of KL is:

$$KL(P,Q) = \Sigma_{i = 1..n} [\ p_i * \log(p_i / q_i) \ ]$$

When we try to compute this formula, we must address two questions:

1. What to do if $q_i = 0$ for some i or $pi = 0$ for some i?
2. How do we define the formula when P and Q are defined over different samples?

The direct answer is that we always consider that: 0 * log (0) = 0.

2.5.3 LexRank

LexRank method for text summarization is another child method of the PageRank method with a sibling TextRank. It uses a graph-based approach for automatic text summarization.

LexRank is an unsupervised graph-based approach for automatic text summarization. The scoring of sentences is done using the graph method. LexRank is used for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences.

In this model, we have a connectivity matrix based on intra-sentence cosine similarity, which is used as the adjacency matrix of the graph representation of sentences. This sentence extraction majorly revolves around the set of sentences with same intend i.e.,

a centroid sentence is selected which works as the mean for all other sentences in the document. Then the sentences are ranked according to their similarities.

2.5.3.1 Graphical Approach

1. Based on Eigen Vector Centrality.
2. Sentences are placed at the vertexes of the Graphs
3. The weight on the Edges is calculated using cosine similarity metric.

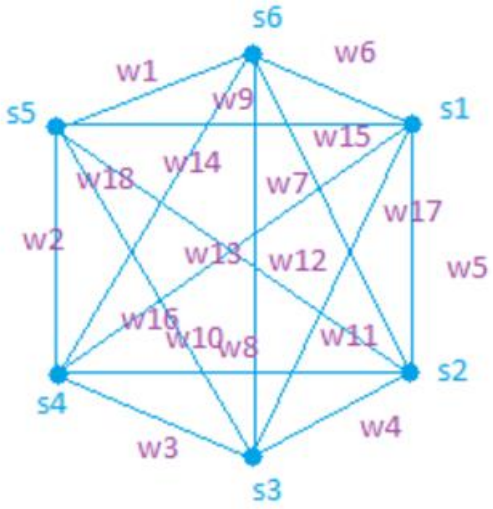


Figure 9: overview of graphical approach

Where Si are the sentences at the vertices respectively and Wij are weights on the edges.

2.5.3.2 Cosine similarity Computation

We order to define similarity; bag of words model is used to represent N-dimensional vectors where N is the number of all possible words in a particular language. For each word that occurs in a sentence, the value of the corresponding dimension in the vector representation of the sentence is the number of occurrences of the word in the sentence times the idf of the word.

$$\text{Idf-modified-cosine}(x,y) = \frac{\sum_{w \in x,y} tfw,x\ tfw,y\ (idfw)^2}{\sqrt{\sum xi \in x(tfxi,x\ idfxi)^2}\ X\ \sqrt{\sum yi \in y(tfyi,y\ idfyi)^2}}$$

Where $tf_{w,s}$ is the number of occurrences of the word w in the sentence s.

$$idfw = \log(\frac{N}{nw})$$

Where N is the number of documents in the collection and nw is the number of documents in which the word w occurs. After all vertices and edges are created, Google's PageRank algorithm is applied to the graph. The idea of applying PageRank is that edges between sentences are votes for the vertices. This creates the idea that highly ranked sentences are similar to many other sentences and many other sentences are similar to a highly ranked sentence. We then create a summary by choosing the highest rated x sentences where x is defined by the user as the number of sentences wanted in the summary.

2.5.4 LSA

2.5.4.1 Definition

Latent semantic analysis (LSA) is an algebraic-statistical method that extracts hidden semantic structures of words and sentences. It is an unsupervised approach that does not need any training or external knowledge. LSA uses the context of the input document and extracts information such as which words are used together and which common words are seen in different sentences. A high number of common words among sentences indicates that the sentences are semantically related. The meaning of a sentence is decided using the words, it contains, and meanings of words are decided using the sentences that contains the words. Singular Value Decomposition, an algebraic method, is used to find out the interrelations between sentences and words.

2.5.4.2 Algorithm

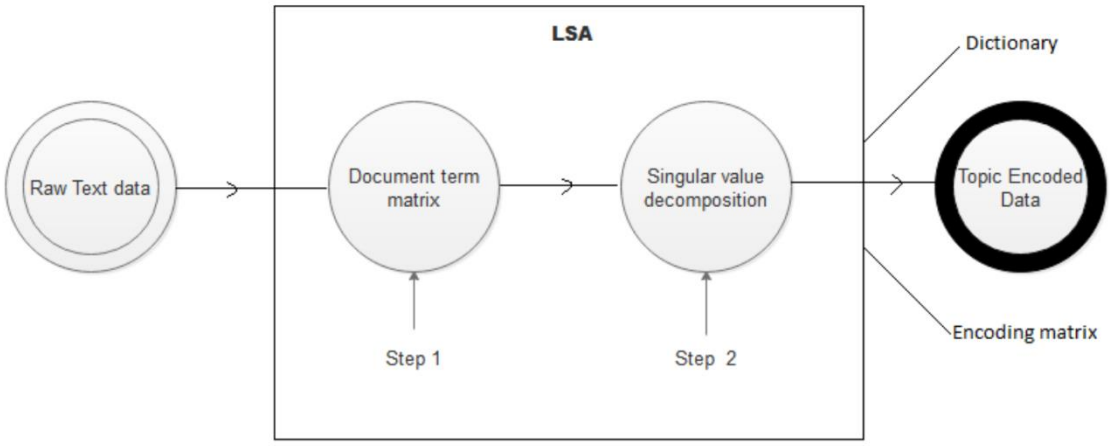


Figure 10: LSA processing

The algorithm for LSA consists of three major steps:

1. **Input matrix creation**: The input document is represented as a matrix to understand and perform calculations on it. Thus, a document term matrix is generated. The cells are used to represent the importance of words in sentences. Different approaches can be used for filling out the cell values. These approaches are as follows.

   • **Frequency of word:** the cell is filled in with the frequency of the word in the sentence.

   • **Binary representation:** the cell is filled in with 0/1 depending on the existence of a word in the sentence.

   • Tf-Idf (Term Frequency-Inverse Document Frequency): the cell is filled in with tf-idf value of the word. Higher tf-idf value means that the word is more frequent in the sentence but less frequent in the whole document. The higher value indicates that the word is much more representative for that sentence than others.

   • **Log entropy:** the cell is filled in with log-entropy value of the word, which gives information on how informative the word is in the sentence.

• **Root type:** the cell is filled in with frequency of the word if its root type is a noun, otherwise the cell value is set to 0.

2. **Singular Value decomposition (SVD)**: In this step we perform the singular value decomposition on the generated document term matrix. SVD is an algebraic method that can model relationships among words/phrases and sentences. The basic idea behind SVD is that document term matrix can be represented as points in Euclidean space known as vectors. These vectors are used to display the documents or sentences in our case in this space. Besides having the capability of modelling relationships among words and sentences, SVD has the capability of noise reduction, which helps to improve accuracy.

In this method, the given input matrix A is decomposed into three new matrices as follows: A = U P VT

3. **Sentence Selection**: Using the results of SVD different algorithms are used to select important sentences. Here we have used Topic method to extract concepts and sub-concepts from the SVD calculations and are called topics of the input document. These topics can be sub-topics, and then the sentences are collected from the main topics.

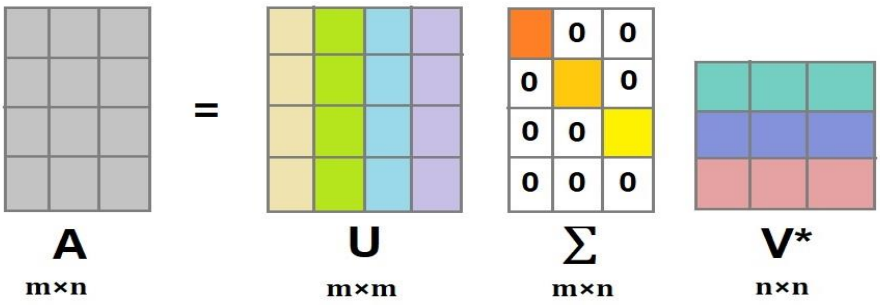


Figure 11: SVD

High number of common words among sentences indicates that the sentences are semantically related. Meaning of a sentence is decided using the word, it contains, and meaning of words are decided using the sentences that contains the word.

2.5.5 LUHN

2.5.5.1 Definition

Luhns Heuristic Method for text summarization. This is one of the earliest approaches of text summarization. Luhn proposed that the significance of each word in a document signifies how important it is.

Luhn's algorithm is an approach based on TF-IDF. It selects only the words of higher importance as per their frequency. Higher weights are assigned to the words present at the begining of the document. It considers the words lying in the shaded region in this graph:

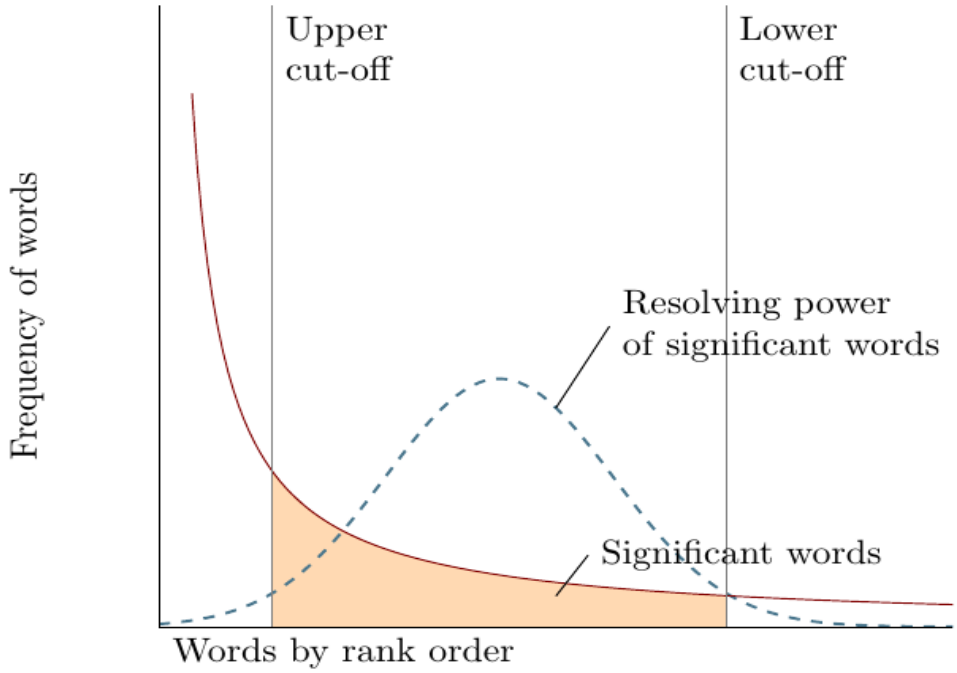


Figure 12: Word-Frequency diagram

The region on the right signifies highest occurring elements while words on the left signifies least occurring elements.

Luhn introduced the following criteria during text preprocessing:

1. Removing stop words.
2. Stemming (Likes->Like).

In this method we select sentences with the highest concentration of salient content terms. For example, if we have 10 words in a sentence and 4 of the words are significant.

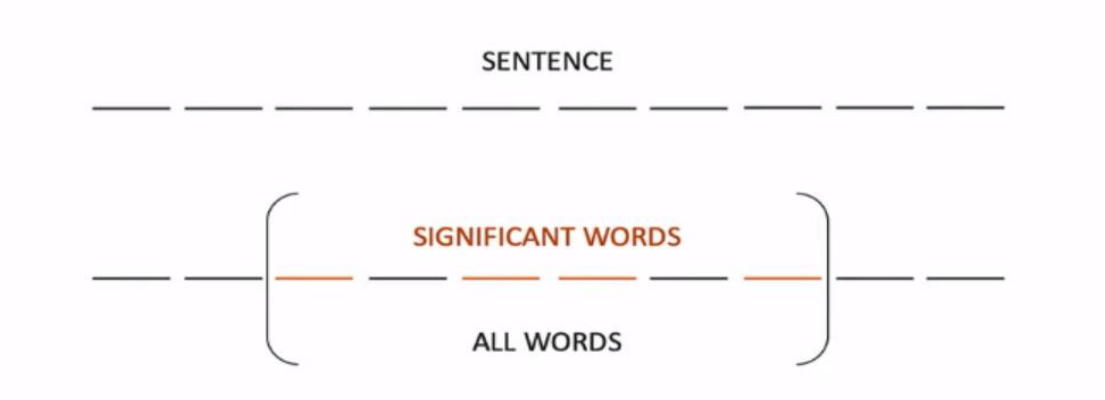


Figure 13: comprehensive example

For calculating the significance instead of number of significant words by all words here are divided by the span that consist of these words. Thus, the Score obtained from our example would be

$$\text{Score} = 42/6 = 2.7$$

2.5.5.2 Algorithm

Luhns method is a simple technique in order to generate a summary from given words. The algorithm can be implemented in two stages.

- **First stage:** we try to determine which words are more significant towards the meaning of document. Luhn states that this is first done by doing a frequency

analysis, then finding words, which are significant, but not unimportant English words.

- **Second stage:** we find out the most common words in the document, and then take a subset of those that are not these most common English words, but are still important.

It usually consists of following three steps:

1. It begins with transforming the content of sentences into a mathematical expression, in other words vector representation (represented below through binary representation). Here we use a bag of words, which ignores all the filler words.

Filler words are usually the supporting words that do not have any impact on our document meaning.

Then we count all the valuable words left to us. For example:

Sentences/Words	INTERN	OPENGENUS	DEVELOPER	ML
An intern at OpenGunus	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

Table 3: Binary representation of sentence

In the above table, we can clearly see that the words like *an* and *a* that are the stop words are not considered while evaluation.

2. In this step we use evaluate sentences using sentence-scoring technique. We can use the scoring method as illustrated below :

Score= (Number of meaningful words) 2/ (Span of meaningful words)

A *span* here refers to the part of sentence (in our case)/document consisting all the meaningful words.

3. Once the sentence scoring is complete, the last step is simply to select those sentences with the highest overall rankings.

2.5.5.3 Problems with the extractive summary

There are many problems with the extractive text summarization feature. These problems can be found in extractive summaries. One of the most common problems have a relation with Extracted sentences. Sentences that usually tend to be longer than average. Secondly, Important or relevant information is usually spread across sentences, and extractive summaries cannot capture this (unless the summary is long enough to hold all those sentences). Finally, Conflicting information may not be presented accurately.

2.6 Conclusion

This chapter concentrates on the extractive text summarization techniques. Thus, automatic text summarization techniques based on an extractive based approach are explained. Then some of English text features have been outlined. Moreover, summarization methods are explained. many points were detailed after dealing with the extractive text summarization. Finally, some problems that people face with extractive summary are mentioned briefly.

Chapter 3

Comparative study of the methods

3.1 Introduction

After making a state of the art on the extractive methods in the previous chapter. This chapter will be divided into two important parts. The first part will be dedicated to the implementation of the extractives methods of automatic text summarization chosen from the state of the art. While the second one is for the ROUGE metric and its implementation. Furthermore, an overview of the development environment will be discussed as a starting point in this chapter, detailing its different tools used.

3.2 Application environment

The implementation and testing of our application was carried out in the following hardware and software environment:

- Processor: Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
- Memory installed (RAM): 6,00 Go
- Windows: Windows 10 Professional
- System type: operating system 64-bit, processor x64
- Python under google colab environment

3.3 Application language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for

use as a scripting or glue language to connect existing components together. Python is simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [9].

3.4 Application software

- **Google colab**

Colaboratory, often shortened to "Colab", is a product (free) of Google Research. Colab allows anyone to write and run the Python code of their choice through the browser. It is an environment particularly suited to machine learning, data analysis and education. In more technical terms, Colab is a hosted Jupyter notebook service that requires no configuration and provides free access to computing resources, including GPUs [10].

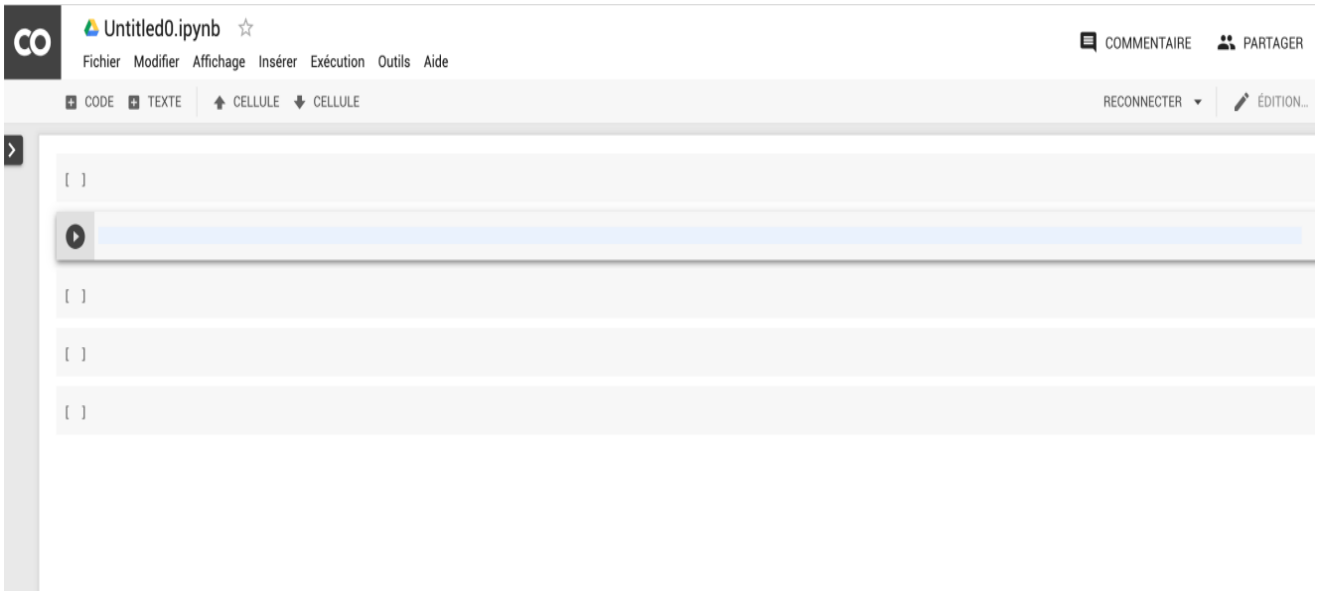


Figure 14: Google Colab interface

- **Jupyter Notebook**

Jupyter Notebook is an open-source web application for creating and sharing documents containing code (executable directly in the document), equations, images and text. With this application it is possible to do data processing, statistical modeling, data visualization, Machine Learning and so on. It is available by default in the Anaconda distribution.

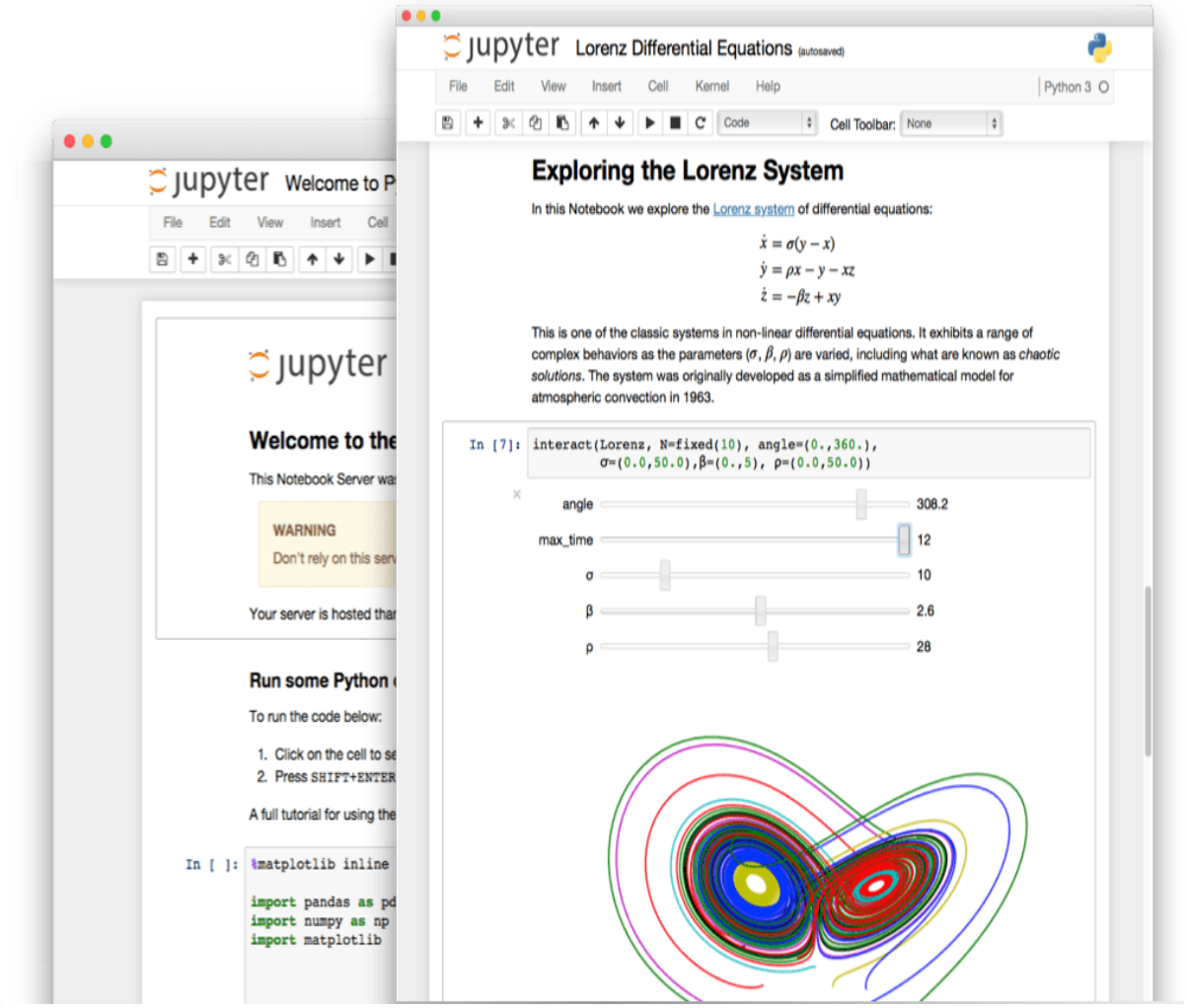


Figure 15: Jupyter Notebook interfaces

- **Google Colab, The cloud version of Jupyter Notebook**

To use Google colab, just go to google drive then click on new then on "more" and choose "Colaboratory".

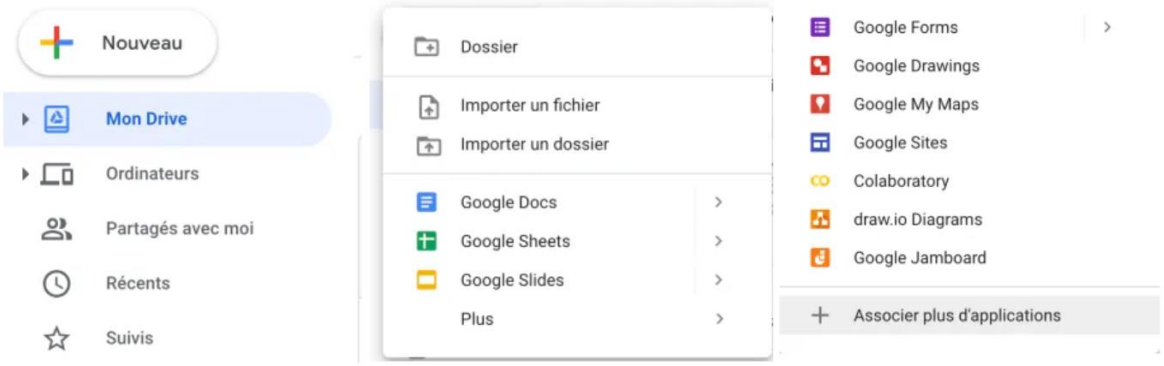


Figure 16: process of creating google colab notebook

3.5 Implementation of methods

Before going directly to the implementation, some tools and packages need to be downloaded and installed as shown in the following caption:

```
!pip install sumy
!pip install nltk
!pip install rouge
!pip install sentencepiece
!pip install bert-extractive-summarizer
```

Figure 17: Libraries downloading/installing

- **! pip install:** is the instruction used for downloading and installing.
- **Sumy:** is a python library for extracting summary from HTML pages or plain texts. Implemented summarization methods as: LexRank, Luhn, LSA, KL-Sum and so on.
- **Nltk:** Natural Language Toolkit is a plateforme used for building python programs that work with human language dater for applying in statistical NLP. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic.
- **Sentencepiece:** is an unsupervised and detokenize used mainly for neural-Network text generation systems where the vocabulary size is pretrained prior to the neural model training. It implements subword units and unigram language model with the extension of direct training from raw sentences. It allows us to make a purely end-to-end system that does not depend on language specific pre/postprocessing.
- **Bert-extractive-summarizer:** This repo is the generalization of the lecture-summarizer repo. This tool utilizes the HuggingFace Pytorch transformers library to run extractive summarizations. This works by embedding the sentences, then running a clustering algorithm, finding the sentences that are closest to the cluster's centroids.
- **Rouge:** this implementation is independent from the “official” rouge script (aka. Rouge-155).

After the installation phase of the packages, we must import them in order to use them.

```
#using sumy package for preprocessing
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer

#using NLTK package
import nltk
nltk.download('punkt')

#using sumy package for LexRank
from sumy.summarizers.lex_rank import LexRankSummarizer

#using sumy package for KLSum
from sumy.summarizers.kl import KLSummarizer

#using sumy package for Luhn
from sumy.summarizers.luhn import LuhnSummarizer

#using sumy package for Lsa
from sumy.summarizers.lsa import LsaSummarizer

#using transformers package,summarizer
from transformers import *
from summarizer import Summarizer

#using rouge package
from rouge import Rouge

#using pandas package
import pandas as pd
```

Figure 18: Libraries importation

3.5.1 LexRank implementation

```
def LexRank(summary):
  # For Strings
  parser=PlaintextParser.from_string(doc,Tokenizer("english"))
  # Using LexRank
  summarizer = LexRankSummarizer()
  #Summarize the document with n sentences
  summary = summarizer(parser.document,21)

  for sentence in summary:
    print(sentence)


  return summary
```

Figure 19: LexRank implementation

For the LexRank method, we created a function based on Sumy package. The function takes as parameter a text to be summarized. In first step the text entered as parameter must go through a preprocessing using PlaintextParser. Then LexRankSummarizer is executed to generate a summary initial. After that, summarizer is executed indicating the number of summary sentences, this summary is returned by the function.

3.5.2 KLSum implementation

```
def KLSum(summary):
  # For Strings
  parser=PlaintextParser.from_string(doc,Tokenizer("english"))
  # Using KL
  summarizer = KLSummarizer()
  #Summarize the document with n sentences
  summary = summarizer(parser.document,21)

  #printing the summarized sentences
  for sentence in summary:
    print(sentence)

  return summary
```

Figure 20: KLSum implementation

For the KLSum method we created a function based on Sumy package. The function takes as parameter a text to be summarized. In first step the text entered as parameter must go through a preprocessing using PlaintextParser. Then KLSummarizer is executed to generate a summary initial. After that, summarizer is executed indicating the number of summary sentences, this summary is returned by the function.

3.5.3 Luhn implementation

```
def Luhn(summary):
  # For Strings
  parser=PlaintextParser.from_string(doc,Tokenizer("english"))
  # Using Luhn
  summarizer = LuhnSummarizer()
  #Summarize the document with n sentences
  summary = summarizer(parser.document,21)

  for sentence in summary:
    print(sentence)

  return summary
```

Figure 21: Luhn implementation

For the Luhn method we created a function based on Sumy package. the function takes as parameter a text to be summarized. in first step the text entered as parameter must go through a preprocessing using PlaintextParser. Then LuhnSummarizer is executed to generate a summary initial. After that, summarizer is executed indicating the number of summary sentences, this summary is returned by the function.

3.5.4 LSA implementation

```
def Lsa(summary):
  # For Strings
  parser=PlaintextParser.from_string(doc,Tokenizer('english'))
  # Using Lsa
  lsa_summarizer=LsaSummarizer()
  #Summarize the document with n sentences
  lsa_summary= lsa_summarizer(parser.document,21)

  for sentence in lsa_summary:
    print(sentence)
  return summary
```

Figure 22: LSA implementation

For the LSA method, we created a function based on Sumy package. The function takes as parameter a text to be summarized. In first step the text entered as parameter must go through a preprocessing using Plaintext Parser. Then lsa_summarizer is executed to generate a summary initial. After that, summarizer is executed indicating the number of summary sentences, this summary is returned by the function.

3.5.5 BERT implementation

```
def BERT(summary):
  model=Summarizer()
  # Specified with ratio
  #summary = model(doc, ratio=0.2)
  # Will return n sentences
  summary = model(doc, num_sentences=21)
  return summary
```

Figure 23: BERT implementation

For the Bert method, we created a function that takes as parameter a text to be summarized. In first step, Summarizer is executed to generate a summary initial. After that, we indicating the number of summary sentences, this summary is returned by the function.

3.6 ROUGE metric

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. It works by comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced).

The main metrics will be covered, ones that are most likely to be used, starting with ROUGE-N.

- **ROUGE-N:** ROUGE-N measures the number of matching 'n-grams' between our model-generated text and a 'reference'. An n-gram is simply a grouping of tokens/words. A unigram (1-gram) would consist of a single word. A bigram (2-gram)

consists of two consecutive words. And higher order n-gram overlap. as the following example shows:

Original: *"the quick brown fox jumps over"*

Unigrams: *['the', 'quick', 'brown', 'fox', 'jumps', 'over']*

Bigrams: *['the quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over']*

Trigrams: *['the quick brown', 'quick brown fox', 'brown fox jumps', 'fox jumps over']*

With ROUGE-N, the N represents the n-gram that we are using. For ROUGE-1 we would be measuring the match-rate of unigrams between our model output and reference. ROUGE-2 and ROUGE-3 would use bigrams and trigrams respectively.

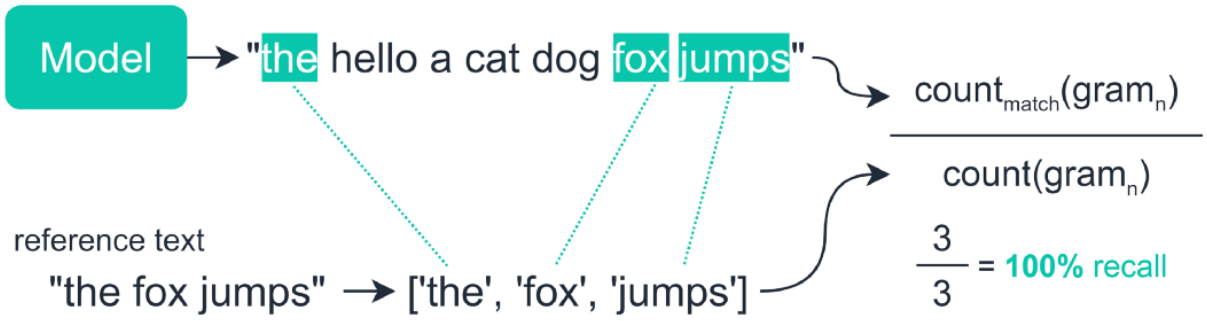
Once we have decided which N to use — we now decide on whether we would like to calculate the ROUGE recall, precision, or F1 score.

Recall: The recall counts the number of overlapping n-grams found in both the model output and reference, then divides this number by the total number of n-grams in the reference. In other words, recall in the context of ROUGE simply means how much of the reference summary, the system summary recovering or capturing it.

$$\frac{number\ of\ n-grams\ found\ in\ model\ and\ reference}{number\ of\ n-grams\ in\ reference}$$

$$\frac{count_{match}\ (gram_n)}{count(gram_n)}$$

In this example, the Recall would thus be:

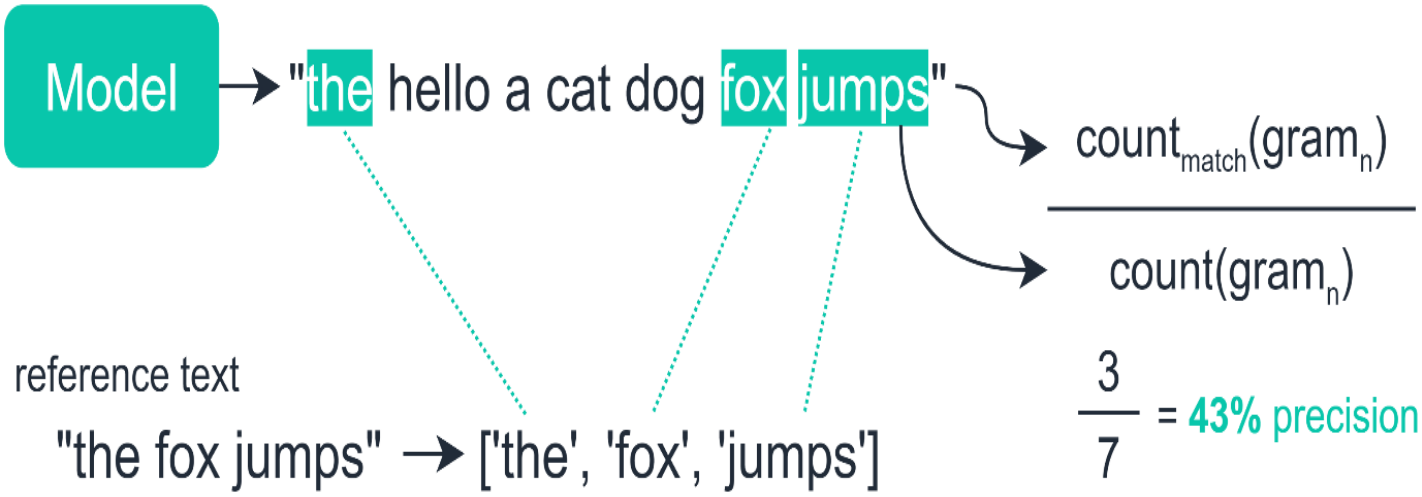


This means that all the words in the reference summary have been captured by the system summary, which indeed is the case for this example. This looks really good for a text summarization system. However, it does not tell you the other side of the story. A machine-generated summary (system summary) can be extremely long, capturing all words in the reference summary. But, much of the words in the system summary may be useless, making the summary unnecessarily verbose. This is where precision comes into play.

Precision: precision is calculated in almost the exact same way as recall, but rather than dividing by the reference n-gram count, we divide by the model n-gram count. Which makes in terms of precision, what you are essentially measuring is, how much of the system summary was in fact relevant or needed? Precision is measured as:

$$\frac{number\ of\ n-grams\ found\ in\ model\ and\ reference}{number\ of\ n-grams\ in\ model}$$

So, if we apply this to our previous example, we get a precision score of just 43%:

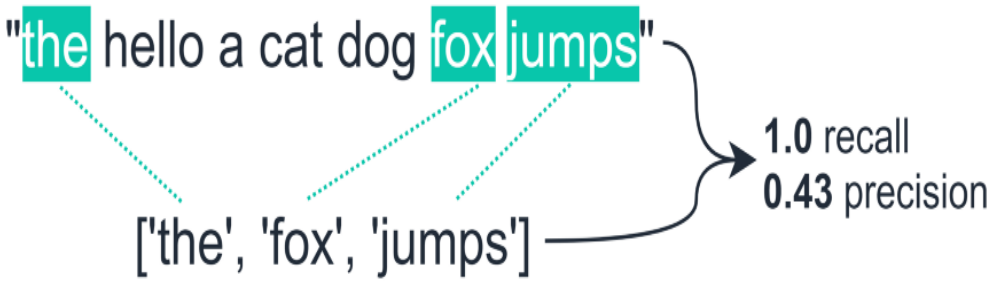


This simply means that 3 out of the 7 words in the system summary were in fact relevant or needed.

F1-Score: Now that we both the recall and precision values, we can use them to calculate our ROUGE F1 score like so:

$$2 * \frac{precision * recall}{precision + recall}$$

Let's apply that again to our previous example:

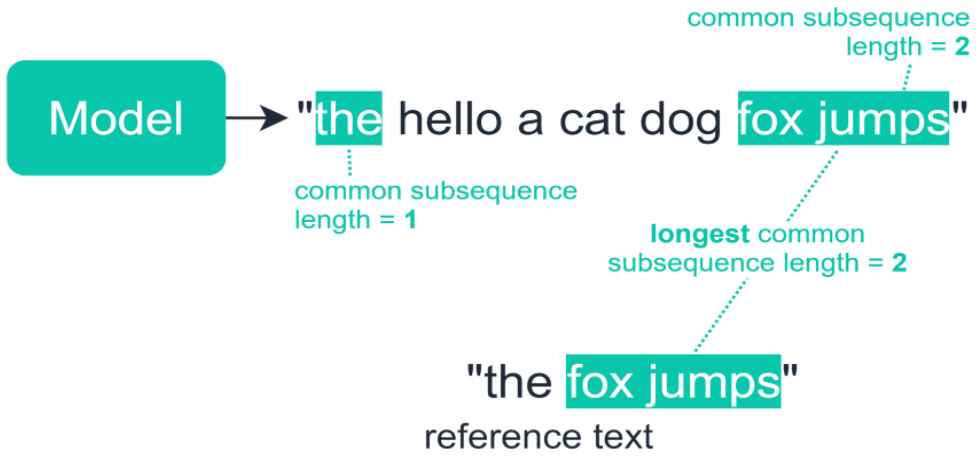


$$2 * \frac{0.43 * 1.0}{0.43 + 1.0} = 0.6$$

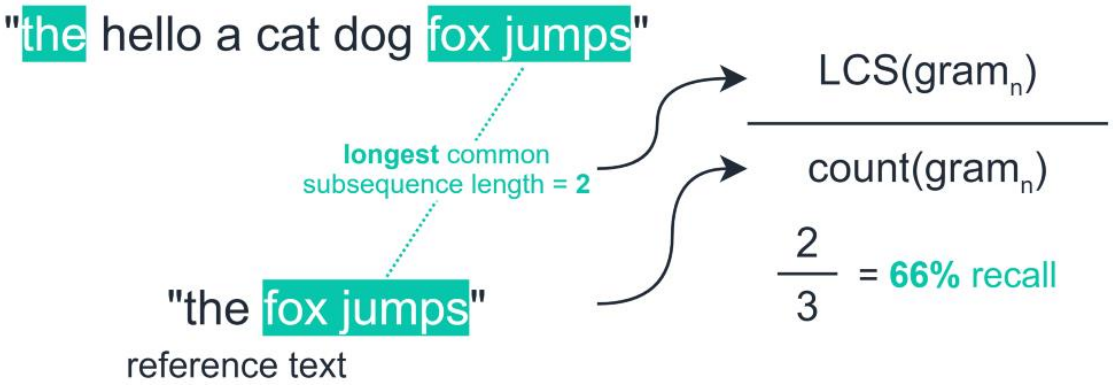
60% f1 score

That gives us a reliable measure of our model performance that relies not only on the model capturing as many words as possible (recall) but doing so without outputting irrelevant words (precision).

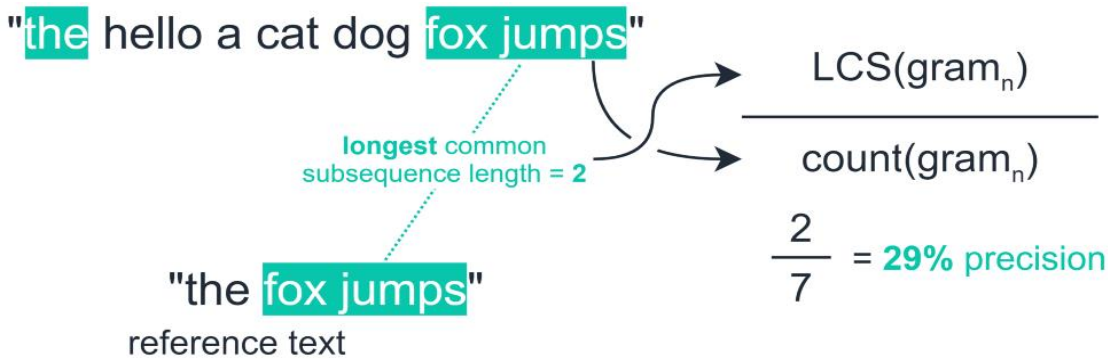
- **ROUGE-L:** measures the longest common subsequence (LCS) between our model output and reference. All this means that we count the longest sequence of tokens that is shared between both:



The idea here is that a longer shared sequence would indicate more similarity between the two sequences. We can apply our recall and precision calculations just like before — but this time we replace the match with LCS:



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



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

$$2 * \frac{0.29 * 0.66}{0.29 + 0.66} = 0.6$$

40% f1 score

- **ROUGE-S:** The final ROUGE metric we will look at is the ROUGE-S, or skip-gram concurrence metric. This metric seems to be much less popular than ROUGE-N and ROUGE-L covered already, but it is worth being aware of what it does. Using the skip-gram metric allows us to search for consecutive words from the reference text, which appear in the model output but are separated by one-or-more other words.

3.6.1 Implementation

Implementing these metrics in Python is incredibly easy thanks to the Python rouge library. We can install the library through pip:

```
[40] !pip install rouge
```

Figure 24: downloading Rouge package

In addition, scoring our model output against a reference is as easy as this:

```
[41] #using rouge package
     from rouge import Rouge
```

```
[42] rouge = Rouge()
```

```
#calculate scores
#LexRank score
score_LexRank=rouge.get_scores(tuple_to_list (LexRank), reference)
#KLSum score
score_KLSum=rouge.get_scores(tuple_to_list (KLSum), reference)
#Luhn score
score_Luhn=rouge.get_scores(tuple_to_list (Luhn), reference)
#Lsa score
score_Lsa=rouge.get_scores(Lsa, reference)
#Bert score
score_Bert=rouge.get_scores(Bert, reference)
```

Figure 25: Calculate scores

The **get scores** method returns three metrics, ROUGE-N using a unigram (ROUGE-1) and a bigram (ROUGE-2), and ROUGE-L.

For each of these, we receive the F1 score **F,** precision **P**, and recall **R**.

3.7 Conclusion

In this chapter, various tools used to implement our application were presented. Moreover, explanation and implementation used for algorithms are chosen and the methods used to compare them were detailed.

Chapter 4

Results

4.1 Introduction

After discussing our environment with its necessary hardware, software and implementing the methods, a comparision between the methods is made by discussing the results and the statistics obtained.

4.2 Experiment

4.2.1 Summarization Datasets

We used the MultiLing 2013 dataset in our experiment; it contains a collection of texts in 40 languages, each text collection contains 30 single text files, and there are about 100–300 sentences in each single text file. In the experiments, we select the data set of the English language as the experimental data set to generate a summary for one single text.

4.2.2 Reference summary

For the evaluation of the summaries generated from the five methods, we created a gold summary or in another term a reference summary of the article, each having an equal number of sentences. The reference summary can be generated either by a human being or by a recognized tool dedicated to this task. In this case it is counted for a tool and not for a human being, the length of the summary is set at 25% of the original text. During the experiment, the total number of the sentences in the original document is 84 and the reference summary consists of 21 sentences according to the agreed rate. The online tool Text Summarizer is chosen to generate our gold summary.

- **Text Summarization:** Text Summarization API is based on advanced Natural Language Processing and Machine Learning technologies, and it belongs to automatic text summarization and can be used to summarize text from the URL or document that user provided [11].

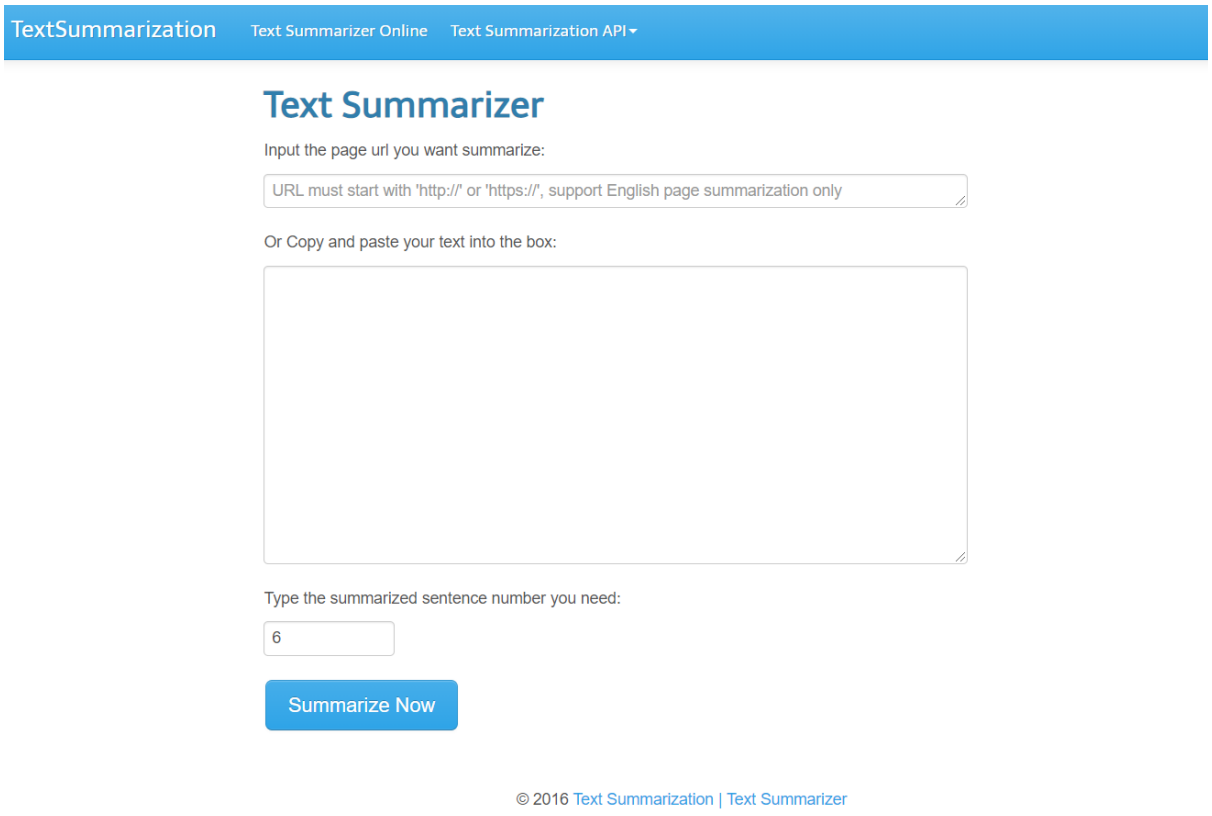


Figure 26: Interface TextSummarization

4.3 Evaluation Method

Experiments are done using the ROUGE tool, which evaluates the performance in terms of three metrics, precision, recall and F-score. In the current study, we have used ROUGE-1, ROUGE-2 and ROUGE-L to evaluate the proposed summarization system. The precision, recall and F-score are adopted for evaluation, which is accomplished with the reference summary.

4.4 Evaluation result

After seeing the different methods for extractive summarization, we take a look at the results obtained from the various implementations done by. The experimental results on the MultiLing 2013 single text dataset are shown in Table 4.

		Rouge-1	Rouge-2	Rouge-L
Lexrank	f	0,496111	0,287446	0,401274
	p	0,432229	0,250377	0,369501
	r	0,582150	0,337398	0,439024
KLSum	f	0,487448	0,280922	0,375671
	p	0,503240	0,290043	0,386029
	r	0,472617	0,272358	0,365854
Luhn	f	0,402920	0,157895	0,292818
	p	0,314709	0,123288	0,242563
	r	0,559838	0,219512	0,369338
LSA	f	0,329273	0,314176	0,439260
	p	0,198739	0,189575	0,285565
	r	0,959432	0,916667	0,951220
BERT	f	0,557875	0,347909	0,473154
	p	0,524064	0,326786	0,456311
	r	0,596349	0,371951	0,491289

Table 4: Experiment results

4.4.1 Analysis and discussion

4.4.1.1 Analysis

From the evaluation results, we will analyze each column independently with its parameters.

- **ROUGE-1:** is used for measuring the overlap of 1-grams.

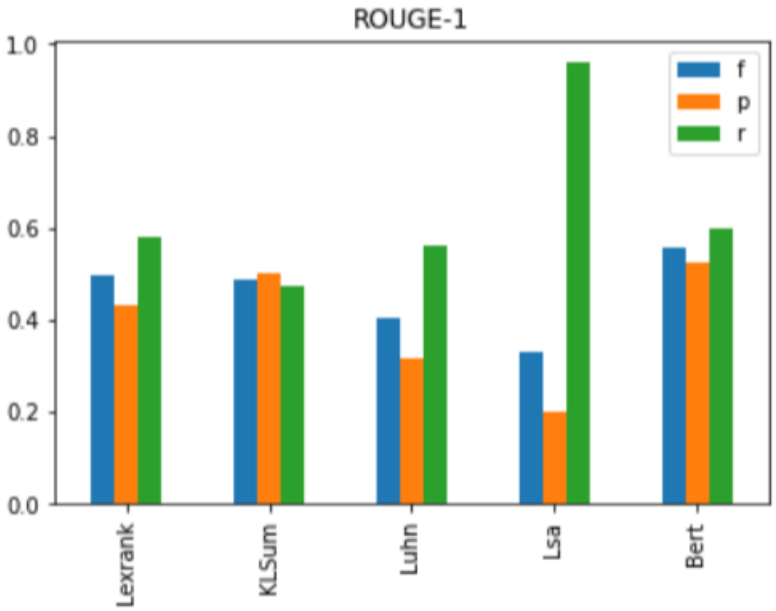


Figure 27: ROUGE-1 results

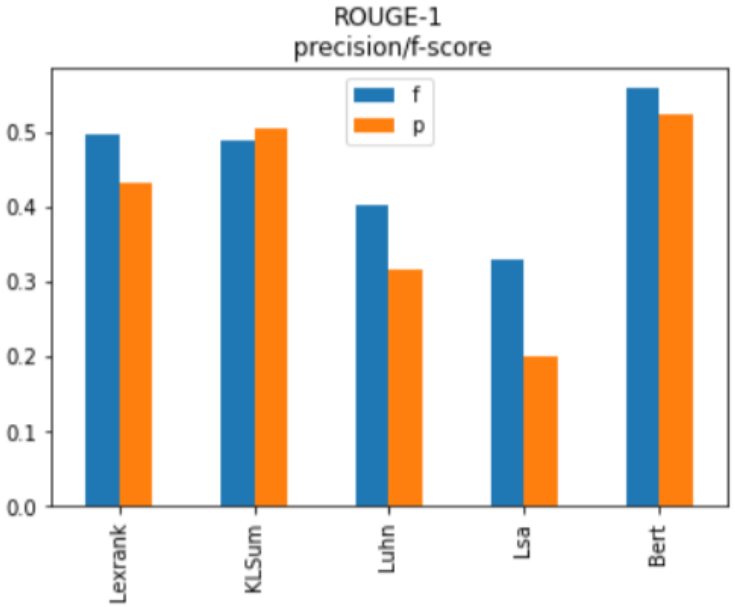


Figure 28: ROUGE-1 (Precision, F-SCORE)

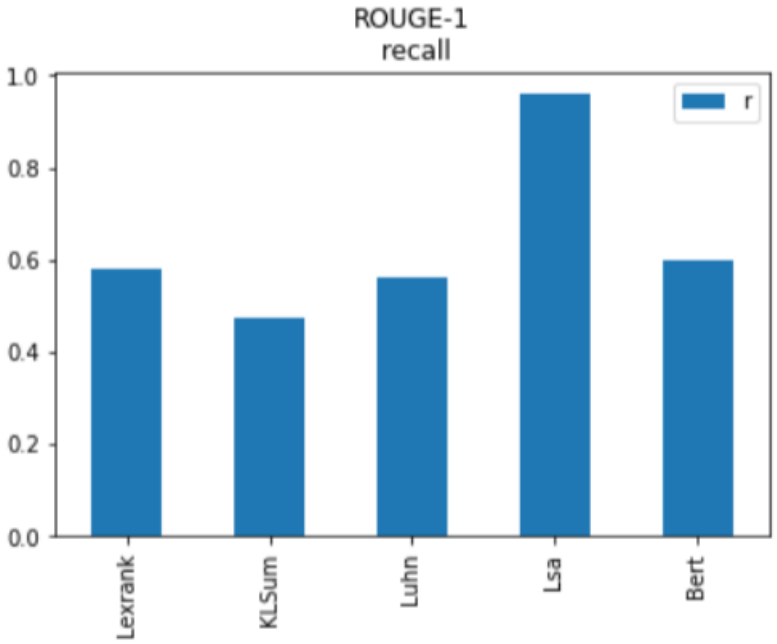


Figure 29: ROUGE-1 (Recall)

- **ROUGE-2:** is used for the overlap of bigrams.

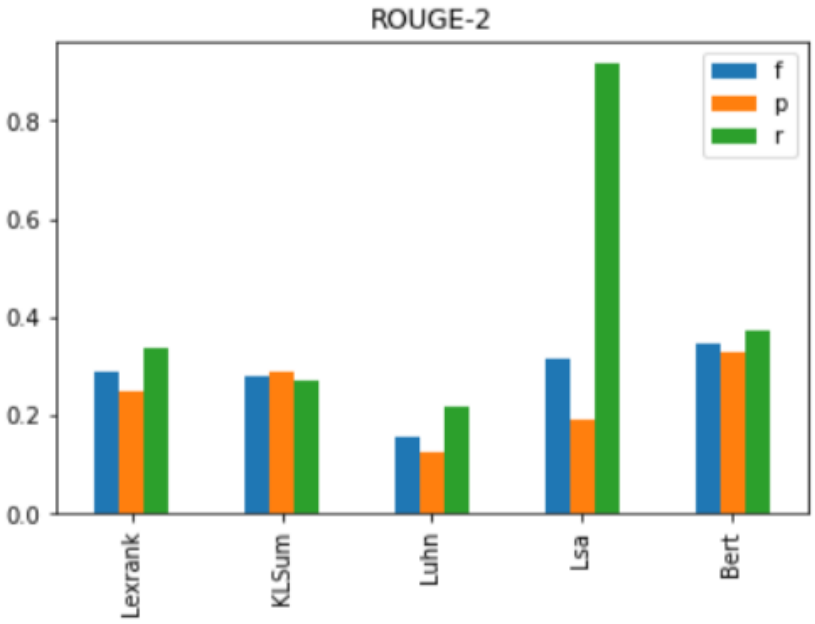


Figure 30: ROUGE-2 results

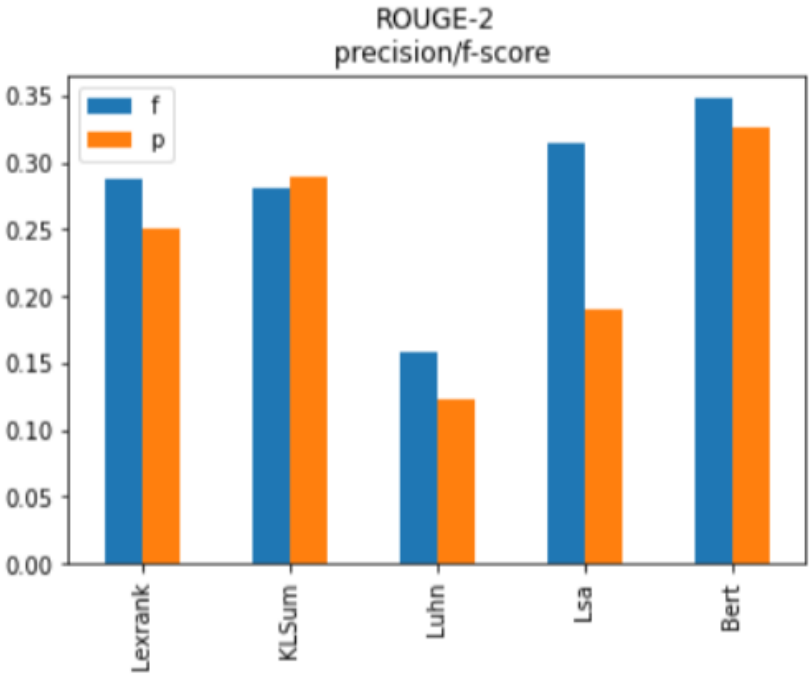


Figure 31: ROUGE-2 (Precision, F-SCORE)

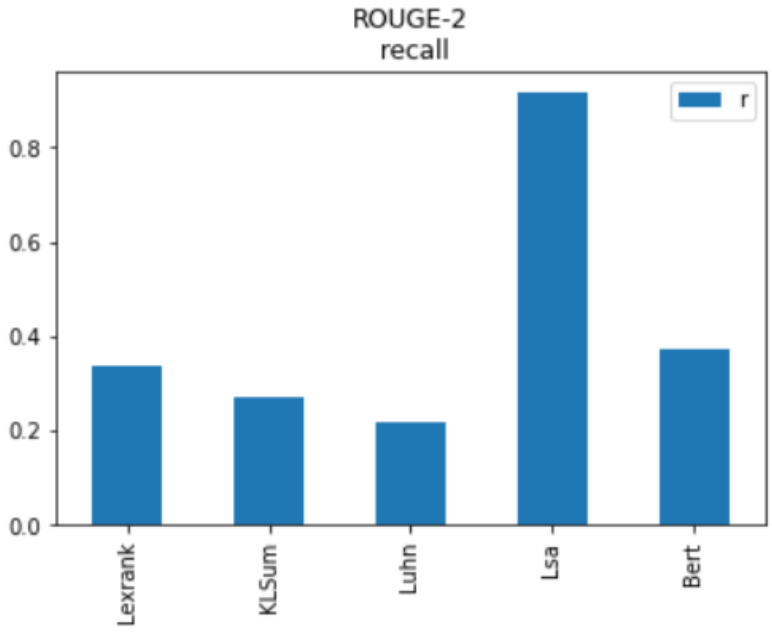


Figure 32: ROUGE-2 (Recall)

- **ROUGE-L**

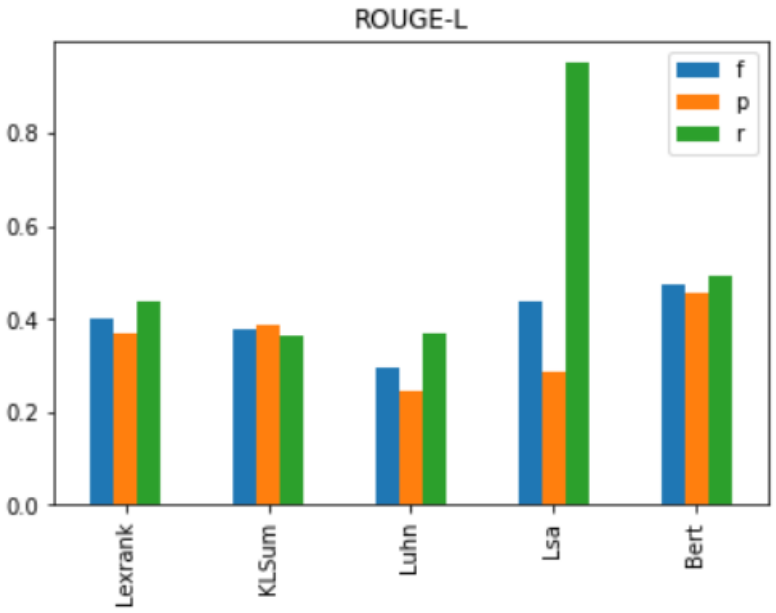


Figure 33: ROUGE-L results

4.4.1.2 Discussion

Based on the analysis of the results obtained and presented in the previous section. We deduce that BERT is the best algorithm among the five in parameter F-score and precision on ROUGE-1, ROUGE-2 and ROUGE-L. Furthermore, LSA algorithm gives the best results in terms of recall parameter in all metrics used.

4.5 Conclusion

This chapter was designed to experiment and bring out results by applying a comparison between the five extractive algorithms. And to show the one that gives the best results. After analyzing and discussing the results, according to our experience we concluded that in terms of the f-score and precision parameters, BERT had the best scores. However, by relying on the recall parameter, it is always LSA that gives the best results.

General Conclusion

Text summarization is growing as sub–branch of NLP as the demand for compressive, meaningful, abstract of topic due to large amount of information available on net. Precise information helps to search more effectively and efficiently. Thus, text summarization is needed and used by business analyst, marketing executive, development, researchers, government organizations, students and teachers also.

So, by the progress in the production of a vast body of information due to the coming of the internet, automatic text summarization drew a significant level of attention to speed the task of providing necessary information for users. However, Different types of summaries such as abstractive, extractive, and single and multi-documents have been explained in this dissertation. Also, numerous summarization techniques like statistical, machine learning, and semantic-based were described as well. The focus was on the extractive summary since abstractive summary needs complex natural language processing method, and this makes it difficult to be studied.

Finally, some evaluation methods were introduced, which could be used to examine and compare the results of different extractive methods. After making the comparison, we came into the conclusion that by utilizing BERT and LSA extractive methods, we can get better results and summaries to a specific document or an article.

Bibliography

Articles

[1] Ilyas Cicekli, Journal of Information Science (2011) Text summarization using Latent Semantic Analysis.

[2] John Frazier and Jonathan Perrier, Text Summarization.

[3] Saeedeh Gholamrezazadeh, Mohsen Amini Salehi, Bahareh Gholamzadeh, IEEE (2009) A Comprehensive Survey on Text Summarization Systems.

[4] AHMED ELREFAIY, AHMED RAFAT ABAS, IBRAHIM ELHENAWY, Journal of Theoretical and Applied Information Technology (2018), REVIEW OF RECENT TECHNIQUES FOR EXTRACTIVE TEXT SUMMARIZATION.

[5] Mehdi Allahyari,(2017) Text Summarization Techniques: A Brief Survey.

[6] Kanitha.D.K ,D. Muhammad Noorul Mubarak, International Journal of Computer Science & Information Technology (2016), AN OVERVIEW OF EXTRACTIVE BASED AUTOMATIC TEXT SUMMARIZATION SYSTEMS.

[7] Diksha Harbola, International Journal for Scientific Research & Development (2018) A Survey on Extractive Text Summarization Techniques.

[8] Arpita Sahoo, Dr.Ajit Kumar Nayak, International Journal of Engineering Research in Computer Science and Engineering (2018) Review Paper on Extractive Text Summarization.

Internet Articles

[9] https://www.python.org/doc/essays/blurb/

[10] https://research.google.com/colaboratory/faq.html

[11] http://textsummarization.net/

Dataset

[12] http://multiling.iit.demokritos.gr/pages/view/1571/datasets George Giannakopoulos (2013)

Books

[13] Ted Kwartler,Text Mining in Practice with R.