uc3m | Universidad Carlos III de Madrid

Master in Big data Analytics

Academic Year 2019-2020

Master Thesis

Extractive Text Summarization: A study of different methods for news articles

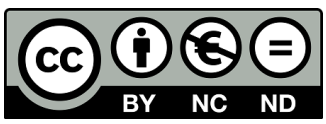
Andrés Escalante Ariza

Advisor: Isabel Segura Bedmar

Madrid, September 2020

AVOID PLAGIARISM

The University uses the **Turnitin Feedback Studio** program within the Aula Global for the delivery of student work. This program compares the originality of the work delivered by each student with millions of electronic resources and detects those parts of the text that are copied and pasted. Plagiarizing in a TFM is considered a **Serious Misconduct**, and may result in permanent expulsion from the University.



SUMMARY

A summary is a shorter version of a longer text. The task of summarizing is complex and time consuming. It can be automated using Natural Language Processing (NLP) to create text summarization algorithms. There are a lot of different methods to create summaries, being the most important options between extractive and abstractive approaches. Extractive methods make a selection of the most relevant sentences of the text and then use them to create the summary, the abstractive methods create new sentences with the main topics of the original text and combine them to form a summary. This paper provides an overview and an implementation guide for different extractive text summarization algorithms. Some of them are more traditional methods and others use machine learning. The methods studied are: i) choosing the first 'k' sentences, ii) a word frequency-driven approach, iii) TextRank, which is a graph based algorithm for keyword and sentence extraction, iv) Latent Semantic (LSA), a topic modeling technique that allows to obtain the most relevant concepts from a text, and v) PreSumm, a machine learning approach that uses a Bidirectional Encoder Representations from Transformers (BERT), one of the most successful recent advances in NLP that has achieved the state-of-the-art performance on multiple NLP tasks. All the background knowledge to understand the idea of each method is explained in this project. Finally a test evaluated with ROUGE, a set of metrics for evaluating text summarization, have been performed to see how these methods compare to each other using different datasets. All the datasets are related to the news articles domain in the English language.

Keywords: Extractive Text Summarization, Natural language Processing, Machine Learning, Deep learning, BERT, ROUGE.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ANN Artificial Neural Network. 5

BERT Bidirectional Encoder Representations from Transformers. 9, 15, 16, 21, 22

CNN Convolutional Neural Network. 7

GRU Gated Recurrent Unit. 15

IDF Inverse Document Frequency. 13

LSA Latent Semantic Analysis. 14, 18, 20, 23, 24, 26

LSTM Long Short Term Memory. 8, 14, 15

ML Machine Learning. 4, 5

MLP MultiLayer Perceptron. 7, 15

NLP Natural Language Processing. 3–5, 19, 22

RNN Recurrent Neural Network. 7, 9, 14, 15

ROUGE Recall-Oriented Understudy for Gisting Evaluation. 11, 12, 14–16, 23, 26

SVD Singular Value Decomposition. 14, 20, 21

TF Term frequency. 13, 19

1. INTRODUCTION

1.1. Motivation

A summary is a abbreviated adaptation of an original document that includes the most important ideas. In order to make one, it is crucial to read and understand the original document, and then extract the main ideas. This process requires intelligence, language knowledge and time. Automatic text summarization aims to leave this task to computers. This task have a lot of use-cases, for example:

- Newsletter: Nowadays users are subscribed to different newsletters, and this tool will help to easily create a short version of the content with no extra work.
- Legal contracts: Not everybody reads all the contracts, a short understandable summary may help the user know what is signing.
- Media monitoring and media response analysis: It can be used to save time for the employees by reducing the amount of data they need to review without losing quality. [1]
- Conferences/videos: They can be transcribed to text and summarized all automatically to create a quick overview of the conference/videos.

Summaries can be classified in different categories.[2]

- Based on the Text to summarize:
  - Single-Document: When the summary is performed from one document, for example a news story or a scientific paper.
  - Multi-Document: When the summary is performed from 2 or more documents, for example, summarizing the news of an event by using different news web-pages trying to avoid redundancy.
- Based on the content of the summary:
  - Indicative: Summary that gives the main idea of the original text to the reader.
  - Informative: Summary made to be read instead of the original text.
- Based on the purpose of the summary:
  - Generic: Summary made for a general audience.
  - Query-based: Summarize the data of the text that is relevant to a specific query.

- Based on how the summary is performed:
  - Abstractive: The summary created is made from new sentences that contain the most important information of the original text [3].
  - Extractive: Selects the most important sentences of the text to create a summary.

Extractive and abstractive summarization offers very different approaches. *"Abstractive summarization methods aim at producing important material in a new way. In other words, they interpret and examine the text using advanced natural language techniques in order to generate a new shorter text that conveys the most critical information from the original text"* [3]. This kind of summary is the most common for people, but it is a difficult task for a computer. Extractive text summarization selects the most relevant sentences of a text. Then creates a summary with these selected sentences. In this way, the summarization task can be considered as a binary classification task for each sentence. Extractive summarization is simpler since there is no need to generate new sentences. They are already written by the text author/s. The next figure shows the difference between extractive and abstractive text summarization.

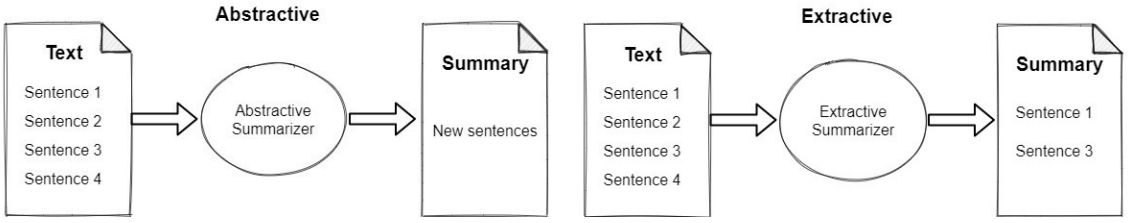


Fig. 1.1. Abstractive vs Extractive approaches

Now on, this document will be focused in Single-Document, informative content, generic purpose and extractive summarization method.

1.2. Objectives

The main objectives of this research are:

1. Know the foundations of the main extractive techniques applied to text summarization.
2. Implement some of these techniques to create a text summarization system.
3. Evaluate and compare them on different datasets for text summarization.

1.3. Structure of the document

Section 2 will give background information. It introduces Natural Language Processing (NLP) and Machine Learning.

Section 3 introduces the main metrics for text summarization.

Section 4 introduces the most common methods of text summarization and some deep learning approaches

Section 5 describes the text summarization datasets used in this research and a more detailed explanation of how different methods have been implemented.

Section 6 shows and discusses the results.

Section 7 gives the conclusions and future directions.

Section 8 will illustrate with a table and a Gantt diagram the steps performed and time in days used to develop this research.

Appendix A gives an example of a news article and the output of the different methods.

2. BACKGROUND

This chapter aims to present the foundations of Natural Language Processing and Machine Learning fields.

2.1. Natural Language Processing

Natural Language Processing (NLP) is a group of techniques used for the analysis and representation of the human language [4]. The goal of these techniques is to represent the human languages in a way that an algorithm can use to produce something with added value [5]. It is used in a wide variety of fields such as machine translation, information retrieval, information extraction, text simplification and text summarization, among others.

In the context of automatics text summarization, NLP is generally used for the processing of the data (text) before making the summary. Depending on the algorithm applied, the prepossessing is different. In general, it involves most or at least a few of the following steps:

- Convert text to lower case.
- Update contractions into its longer versions such as "I' m" into "I am" .
- Remove special characters such as "@".
- Remove the stop words that do not add information (for example "the", "that").
- Split the text in sentences.
- Divide sentences in tokens.
- Creation of matrices that represent the text by encoding the words. To perform this, word embedding models are used. *"Distributional vectors or word embedding essentially follow the distributional hypothesis, according to which words with similar meanings tend to occur in similar context. Thus, these vectors try to capture the characteristics of the neighbors of a word."* [6]. They are normally computed with pretrained deep neural networks [7] [8] that assign a numeric value (vector) to each word in a low dimensional space.

2.2. Machine Learning

Machine Learning (ML) is born with the idea to solve a problem in which there is no set of general instructions to solve it. The idea is to use examples to make the computer

"learn" and deliver a solution to the problem by creating a model. Depending on the data used, machine learning algorithms can be classified between supervised or unsupervised algorithms. Supervised it is referred when the data have a known solution. In other words, in a supervised classification problem during the training stage there is a supervisor that tells which is the right answer which helps to learn. In unsupervised, the data does not have a known solution. [9]

Machine Learning requires 3 different stages: data mining/acquisition, training and prediction.

The input data for the training is very important and different data can give very different results. The data mining stage consists in acquiring the data to train the model. In general, the input data is divided into training data and test data.

The training stage uses the input data to create a model that predicts the output The training data is used to create and optimize the model, while the testing is to evaluate the trained model and have an idea of the performance.

Finally, the prediction stage gives desired output. If the model input data was representative and the model have been created accordingly, the final outputs should expect similar results to the one in the test dataset.

One of the main problems in ML is overfitting. This is a phenomena where the model provides very high performance in the training stage, but with low results in the prediction stage. The model is very specific to the input data and has memorized the input data without generalizing the problem, giving bad predictions.

There are several supervised classification algorithms to create models such as K-nearest neighbors (kNN) [10], Logistic regression [11], Support vector Machines [12] or Neural Networks [13]. In the last years, deep neural networks have brought a revolution in ML and in NLP, achieving the state-of-the-art results for many tasks in these fields.

2.2.1. Artificial Neural networks (ANN)

Artificial Neural Networks are inspired by the neurons in out brains. A Perceptron is a building block of an ANN. It consists of x binary inputs and its weights to produce a single binary output [13].

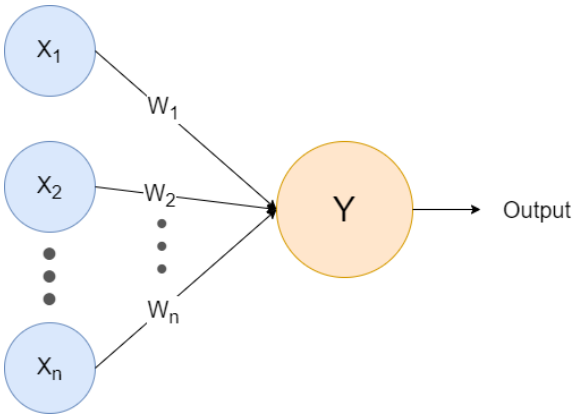


Fig. 2.1. Perceptron Schema

The binary output is decided by the following formula:

$$f(x) = \begin{cases} 1, & \text{if } \sum_n w_n x_n \geq threshold \\ 0, & \sum_n w_n x_n \leq threshold \end{cases}$$

However, the concept of a perceptron can be further expanded by changing the formula used to calculate the output (which is called the activation function). There are a lot of activation functions besides the step function that uses the perceptron, for example:

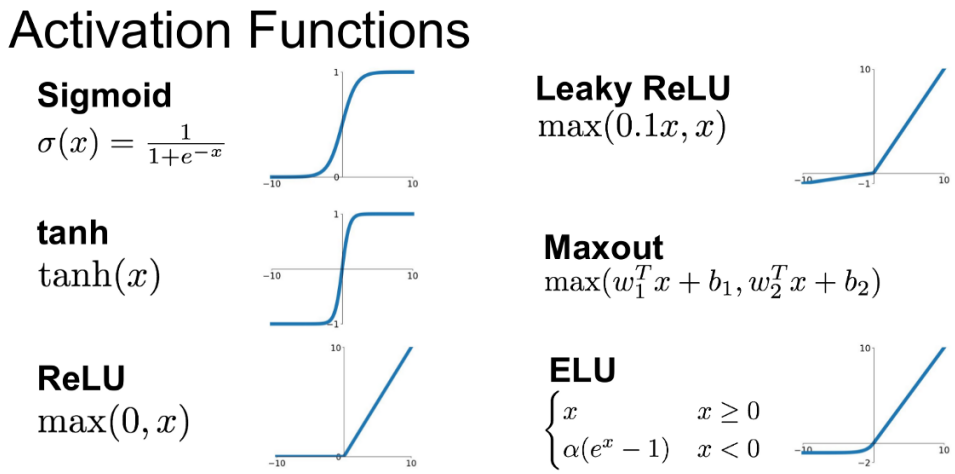


Fig. 2.2. Activation functions. This figure has been taken from [14]

Neural networks are made of these perceptrons (or variations of it). The networks are normally initialized with random weights or as a Gaussian distribution of mean 0 and standard deviation 1. They can be also initialized using pretrained models (for example, word embedding models). At each iteration, a loss function is calculated to grade the model, such as quadratic error or cross-entropy cost function. The weights are calculated using gradient descent and backpropagation with a learning rate big enough to make meaningful changes. The objective of training a model is to minimize the loss function.

Backpropagation is the technique to calculate the gradients and gradient descent is the method used to update the parameters with the objective to obtain a lower value in the loss function. [15]

The networks can have different architectures for example MultiLayer Perceptron (MLP) [16],Convolutional Neural Network (CNN) [17] and Recurrent Neural Network (RNN) [18]. Due to the sequential nature of text, RNN architecture is the best suited for NLP tasks. The recurrent connections ensure that the sequential information is captured.

RNN Architectures

There are different RNN architectures, depending on the input and output data.

- Sequence to vector: It receives a sequence, for example a text as input and the output is a vector for example a number indicating probability.
- Vector to sequence: The input are vectors and the output is a sequence
- Sequence to sequence: Both the input and outputs of this type of architecture are sequences

Each architecture is best suited for a certain task. For example sequence to vector can be used for sentiment analysis where the input is a sentence and the output is probability of being a positive sentiment sentence. Sequence to vector can also be used in extractive summarization, since the output can be a vector with the decision, if the sentences would be included in the summary or not. Vector to sequence can be used to create sentences with certain parameters, for example the parameters can be the number of words, a language and the sentiment, the output would be a sentence with those specifications. Finally sequence to sequence can be used to translate text, where the input is a text in one language and the output the same text in another language. Sequence to sequence can also be used in text summarization. The input would be the full text and the output would be the generated summary.

It is important to notice that in text summarization the output is not of a fixed length, and that it is different from the input size. To be able to have variable output an Encoder-Decoder Architecture [19] can be used. It uses two RNNs, one as encoder and other as decoder. First, the encoder converts the input sequence to a vector. Then, the decoder transforms the vector to a sequence. In this way, the architecture is able to deliver variable lengths outputs. For text summarization, the input would always be larger than the output but the length in words of the output is unknown.

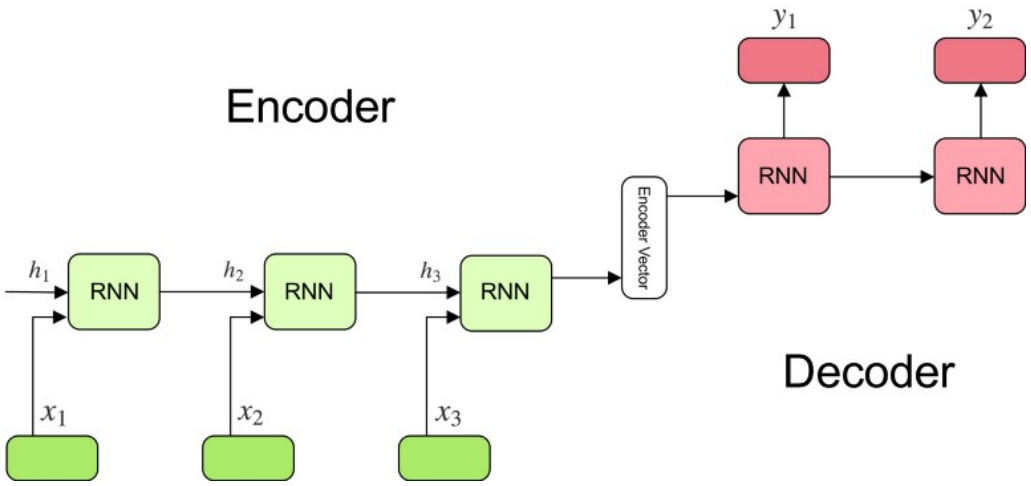


Fig. 2.3. Example of Encoder Decoder Architecture. This figure was taken from [20]

In figure 2.3, the x_n are the words of the sentences. The encoded vector is the word embedding discussed in Section 2.1. Finally, the y_n represents the words of the summary.

When the architecture has to process very large sequences, there are problems called the vanishing and exploding gradient. In these problems, the gradients calculated tend to 0 or infinity, meaning that the input information could be lost. To avoid this problem, a Long Short Term Memory (LSTM) can be used. It was proposed by Hochreiter and Schmidhuber in [21] and it uses a LSTM cell. Each cell has 3 gates, input, forget and output gate, to control the information that propagates forward. This avoids the vanishing and exploding gradient problems for longer networks. The following image describes the architecture of a LSTM cell:

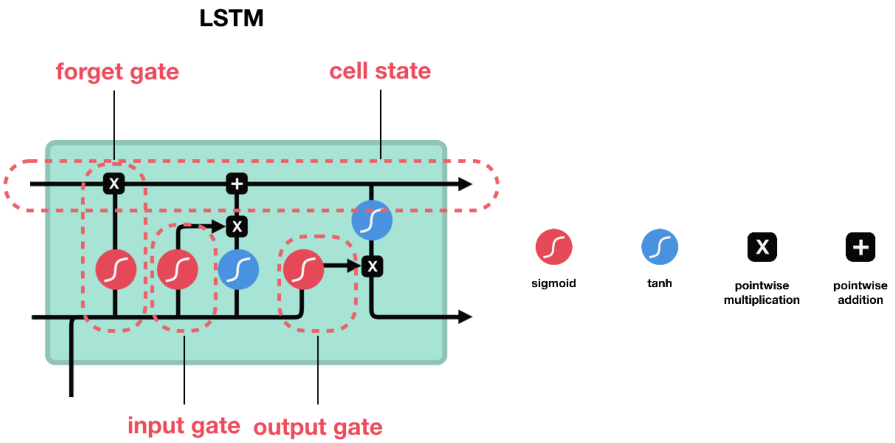


Fig. 2.4. LSTM Architecture [22]

LSTM can be expanded with an attention mechanism. It uses the output of each LSTM cell to learn to pay selective attention to these states and relate them to the output sequence.

Transformers introduced in [23], expand this concept of attention. *"The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution."* [23], understanding by transduction as the transformation of input sequence to output sequence.

2.2.2. BERT

BERT stands for Bidirectional Encoder Representations from Transformers [24]. It is a model created by Google, which has successfully been used in many NLP tasks. As its name suggests, it is a bidirectional encoder that uses the previously introduced transformers. It analyses the text in both ways, left to right and right to left, which helps to understand the context of the text. BERT has been trained in two steps: Pre-training and Fine-Tuning. The idea is to use the pre-trained version and add an additional layer to create state of the art models. [24]

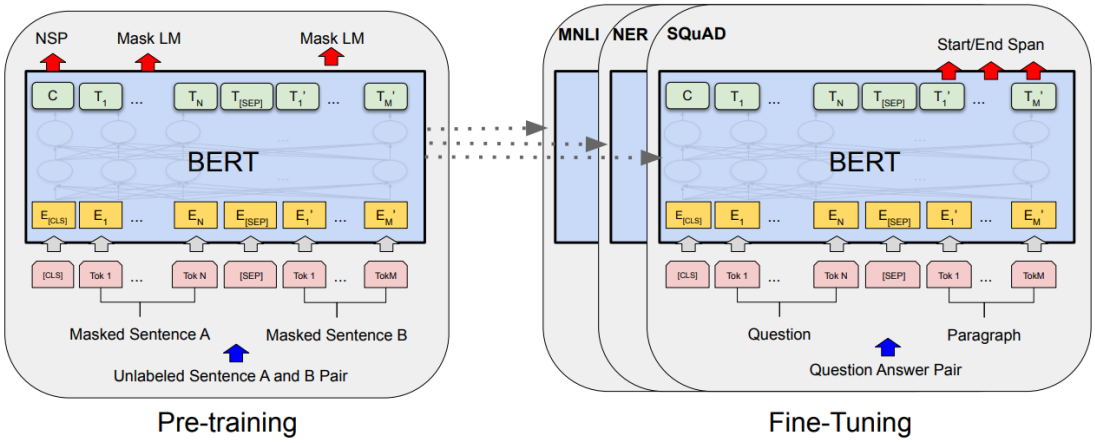


Fig. 2.5. Pre-training and Fine-tuning procedures [24]

In the Pre-training stage, the first task consists of training a Masked LM (MLM) model. From the input data, some tokens are masked (instead of putting the word, a mask token is used) and then predict those tokens. The second task is Next Sentence Prediction (NSP), the relation of two sentences is learned. This corresponds to the left part of figure 2.5. It is trained using BookCorpus [25] (with 800M words) and English Wikipedia (with 2500M words, only text).

The right part of Figure 2.5 is an example for the Fine-tuning stage, in this case for a question answer purpose. However, this task can be changed for other purposes.

BERT can handle different inputs. The first token is `[CLS]`, a special classification token. Then, it expects normal words as a token. If there are more than one sentences, they have to be separated by the special token `[SEP]`. Figure 2.6 shows a representation of the input that BERT expects and how the embeddings are calculated.

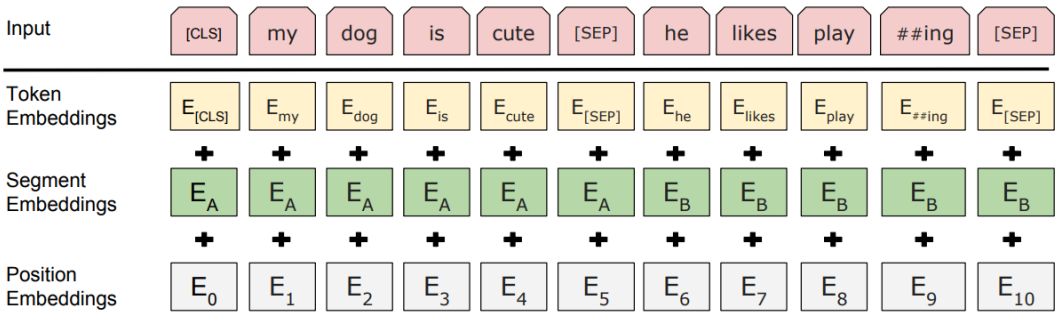


Fig. 2.6. *"BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings."* [24]

3. EVALUATION FOR TEXT SUMMARIZATION

The evaluation of summaries is a difficult task. Traditionally in schools, the teachers have evaluated summaries in a subjective way. For example, the evaluation criteria of the summarization exercise in the exam to access universities (Comunidad de Madrid, Spain), says: *"The summary should contain the principal idea of the text exposing them with internal coherence"* [26]. Ideally, a human should look to all the summaries created and grade them. This method works well in a school environment, however, it requires the teacher to read the original article and all the generated resumes. The main weakness of this evaluation approach is that it is very time-consuming. Moreover, it would be very difficult to compare different methods across different studies. Thus, this evaluation method is very subjective and difficult to replicate. Moreover, different people might grade differently.

Theoretically, the classification nature of extractive text summarization gives the option to use regular ways of evaluation used in binary classification, for example the confusion matrix, precision, recall or F1.

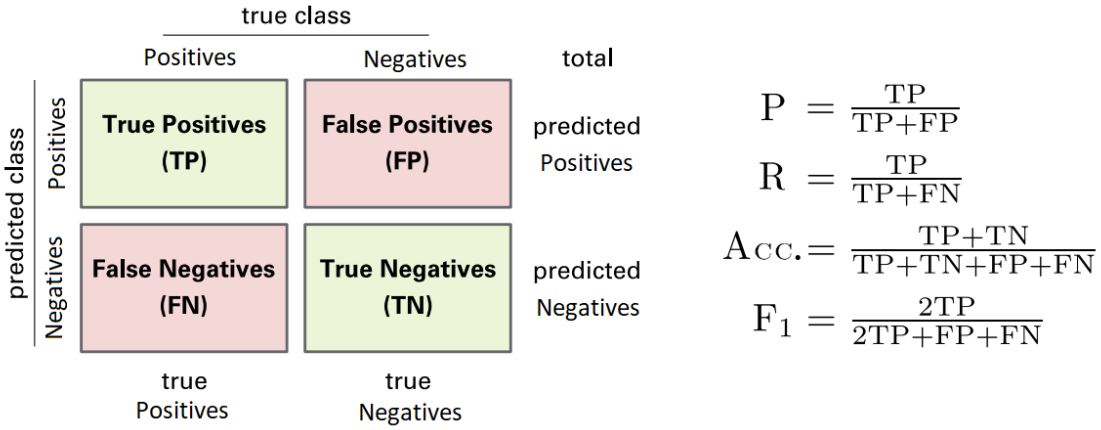


Fig. 3.1. Confusion Matrix with Precision (P), Recall (R), Accuracy (Acc.) and F1 Formulas, This figure was taken from [27]

However, the datasets for text summarization do not provide the answer in an extractive manner (where each sentence should be labeled to be or not included in the final summary). In these datasets, the answer available is an abstractive summary, created by a human. For this reason, the evaluation has to be done with a different approach.

The most common method of evaluation text summarization is Recall-Oriented Understudy for Gisting Evaluation (ROUGE). It is an objective and fast metric created by Lin, Chin-Yew in [28]. It requires to have an hypothesis (the automatic generated summary) and a reference (a human generated summary) from the same source text. ROUGE measures the similarity between texts. It uses the words (or n-grams).

There are different variations of ROUGE. The most commonly used and the ones used here are ROUGE-1, ROUGE-2 and ROUGE-L, which are calculated using the following formula:

$$Rouge - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

[28]

"Where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries" [28].

ROUGE-1 uses 1 gram, ROUGE-2 uses 2 gram and ROUGE-L uses the largest common sub-sequence. For example a ROUGE-N score of 0.3 can be interpreted as that 30 % of the n-grams in the automatic generated summary are also in human generated summary. This score as the name ROUGE suggests is a recall score. A precision score can be calculated equally by swapping both inputs in the formula. The scores described below use f1-score using both the precision and recall ROUGE scores. Which is an approach very commonly used across different papers. It follows the next formula:

$$Rouge\ F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

The author of ROUGE concluded in [28] that ROUGE-1 and ROUGE-L performed good for very short summaries and that ROUGE-2 and ROUGE-L performed good for single document summarization tasks.

4. EXTRACTIVE METHODS FOR TEXT SUMMARIZATION

This chapter describes the most common techniques for extractive text summarization. It also includes the description of the metrics used to evaluate the text summarization systems.

4.1. Word Frequency approach

This method uses the frequency of each word to score sentences. The score of each sentence is calculated by adding the score of each of its words. Then the highest scoring sentences are used to create the summary. There are different approaches for scoring the words. [29]

- Term frequency (TF): It is calculated by counting the number of times each word appears in the complete text.
- Inverse Document Frequency (IDF): It is an extension of the previous method for multiple documents. To calculate the scores it takes into account the frequency of each word as well as if it appears in all documents. The formula is:

  $IDF = log(\frac{N}{n_j})$

  Where N is the number of documents and n_j the number of documents where the word j appears. It can be use in single document if each sentence is considered as a document. Then N is the number of sentences and n_j would be the number of sentences where the word j appears
- TF-IDF: is a combination of the previous two methods of scoring which gives more importance to less frequent words that are very frequent in one document/sentence. The formula is:

  $TF - IDF = TF \cdot IDF$

[29]

4.2. Graph theory approach

This approach uses graph theory to select the sentences. It has two important steps, the creation of the graph and then the identification of its most important nodes. To create the graph, each sentence is represented as a node. Then, if two sentences are similar, an edge between their nodes is created. To measure the semantic similarity, different techniques can be used such as cosine distance or counting the number of common words in both

sentences. Then, it can be used common graph theory algorithms to identify the best nodes (sentences), such as HITS [30] or PageRank [31]. Finally create the summary with the top scoring sentences.

4.3. Latent Semantic Analysis (LSA)

Latent Semantic Analysis is a technique that attempts to look for semantic related sentences by creating a matrix of the text and then utilizing Singular Value Decomposition (SVD) to find principal orthogonal dimensions of the matrix. It can find closely semantic sentences even when they do not share common words. The selected sentences tend to be relevant and non-redundant due to the characteristics of SVD [32].

4.4. Deep learning methods

Extractive text summarization can be considered as a classification problem, and thereby, it can be approached with machine learning classifiers. These can be approached from different perspectives, where the more successful are the ones using deep learning. The web http://nlpprogress.com/english/summarization.html shows the progress of the text summarization task for different datasets and deep learning methods. This section focuses on the official extractive methods that have been tested in the Non-Anonymized Version of CNN / Daily Mail dataset [33] [34]. It is a widely used dataset of news articles, it is Non-Anonymized because it has not been preprocessed to change named entities as The United Nations, with its own unique identifier e.g., @entity5.

There are more methods that use other datasets, however this dataset is the only one related to extractive summarization for news articles which is the focus for this study. These methods are the state of the art in extractive text summarization for news articles.

4.4.1. Ranking Sentences for Extractive Summarization with Reinforcement Learning

It uses a Sentence encoder, document encoder and sentence extractor with reinforce learning (rewards good summaries at uses to update the model). It tries to optimize the Rouge score directly instead of using a loss function as Cross-Entropy . It uses Recurrent Neural Networks with LSTM cells and a softmax layer (to make the final decision) [35].

This method archived in June 2018 a state-of-the-art result (ROUGE-1: 0.4, ROUGE-2: 0.182, ROUGE-L: 0.366) in the Non-Anonymized Version of CNN / Daily Mail dataset.

4.4.2. BANDITSUM: Extractive Summarization as a Contextual Bandit

This work [36] is based on reinforced learning, where an agent (called bandit) makes a decision and receives a reward based on that decision. A contextual bandit is very similar, but before every decision, a context is shown to the agent, then the reward depends on the decision and the context. In this way, the goal is to quickly learn the best actions for different contexts. This can be interpolated to extractive text summarization by taking each document as context and each sentence of the document as an action (if it would be included as summary or not). The reward function uses a function where the ROUGE score is a main component. The neural networks uses a encoder - decoder architecture, where the encoder is a bidirectional RNN and the decoder is a MultiLayer Perceptron (MLP) using sigmoid activation in the last layer. [36]

BANDITSUM achieved a result close to the state-of-the-art result (ROUGE-1: 0.415, ROUGE-2: 0.187, ROUGE-L: 0.376) in the CNN / Daily Mail dataset in October 2018.

4.4.3. Neural Document Summarization by Jointly Learning to Score and Select Sentences (NeuSUM)

This method aims to do the scoring and selection of sentences as one step. The benefit of scoring and selecting in the same step is to be aware of previously selected sentences. It is necessary to define the number of sentences that the summary would have. This system uses a hierarchical document encoder with two levels: the sentence level and the document level, and a sentence extractor. The sentence extractor scores the encoded sentences and selects the higher scoring sentences until the predefined number of sentences is reached . The encoder is made by Bidirectional GRUs (Gated Recurrent Unit) and the joint scoring and selection uses a GRU as a recurrent unit to remember partial information and a two layer MLP. [37]

NeuSUM achieved state-of-the-art results (ROUGE-1: 0.4159, ROUGE-2: 0.1901, ROUGE-L: 0.3798) in the CNN / Daily Mail dataset in July 2018.

4.4.4. PNBERT

In [38], the authors made a study using different architectures and learning schemes. They explored how using LSTM or transformers affect the final performance as a document encoder. For the decoder, they tested how the final score was affected by sequence labeling (SeqLab) and Pointer network [39]. Sequence labeling is the standard approach of giving a binary output to each sentence selecting or not to be included in summary. Pointer network is similar but applying an attention mechanism that makes it aware of the previous selected sentences. They also tested the influence of using external knowledge with the use of pretrained models such as BERT [24]. The conclusions showed that using BERT as word embedding increases the performance in all architectures. Moreover, LSTM +

Pointer network + BERT + reinforced learning archived state of the art results (ROUGE-1: 0.42.69, ROUGE-2: 0.1960, ROUGE-L: 0.3885) for the CNN / Daily Mail dataset in July 2019 .

4.4.5. Text Summarization with Pretrained Encoders (PreSumm)

It uses a modified version of BERT as an encoder. Then there are 3 different variants of this model. The abstractive version that uses the encoder decoder architecture. A combination of extractive and abstractive that first trains the model encoder (that uses BERT as a start) in an extractive manner and then fine-tuned as abstractive. And an extractive variant, that after the encoder uses several inter-sentences transformers layers. The lower transformer layers capture features of the neighbors sentences while the higher level layers with the self attention that characterize transformers capture document level features [40].

This model obtained state of the art results (ROUGE-1: 0.4385, ROUGE-2: 0.2034, ROUGE-L: 0.3990) for the CNN / Daily Mail dataset in September 2019.

4.5. Extractive Summarization as Text Matching (MatchSum)

This paper approaches the extractive summarization as a semantic text matching problem. The data is transformed to a semantic space. The idea is to rank the sentences of the text and create different extractive summaries with the highest scoring sentences. Then transforms the text and the proposed summaries to a semantic space. Intuitively in this semantic space the closer to the document the better the summary. For the transformation to the semantic space it is proposed a Siamese-BERT architecture consisting of 2 BERTs with tied-weights. Siamese refers to the use of two parallel models that are later joined by a connection function. The connection function used is cosine-similarity. The summary selected is the closest to the document in the semantic space [41].

This method obtained state of the art results (ROUGE-1: 0.4441, ROUGE-2: 0.2086, ROUGE-L: 0.4055) in April 2020 for the CNN / Daily Mail dataset.

5. METHODOLOGY

This chapter describes the datasets and the main extractive techniques explored in this research. The code used and a depth description at each step can be found in https://github.com/Andres-Escalante/Extractive-Textsummarization.

5.1. Datasets

The datasets used in this research are all related to news articles in the English language but from different sources. Each dataset consists of a compilation of stories (news articles) and their corresponding summaries, these summaries have been made by humans.

The first dataset is composed of news from Hindu, Indian times and Guardian which can be downloaded in the following link: https://www.kaggle.com/sunnysai12345/news-summary or https://github.com/sunnysai12345/News_Summary. From now on it would be called as News-summary dataset.

The second dataset is build of news from the Non-Anomalized version of the CNN-Daily Mail dataset and can be found in the following link: https://cs.nyu.edu/~kcho/DMQA/ or downloaded from tensor-flow in https://www.tensorflow.org/datasets/catalog/cnn_dailymail [33] [34]. From now it would be called as CNN / Daily Mail dataset.

The third dataset consists of news from newser.com. It can also be found in tensorflow in https://www.tensorflow.org/datasets/catalog/multi_news. [42] From now on it would be called as Multi-news dataset.

The datasets have been divided in training, validation and test datasets. The training and validation datasets are used to train the models and perform hyper-parameter-tuning. The test dataset is used to obtain the final results.
The following table makes a comparison of the number of instances as well as the train, validation and test splits of the datasets:

	Number of instances	Train split	Validation split	Test split
News-summary dataset	98,400	84,400	7,000	7,000
CNN / Daily Mail dataset	311,971	287,113	13,368	11,490
Multi-news dataset	56.216	44,972	5,622	5,622

Table 5.1. COMPARISON OF NUMBER OF INSTANCES OF THE DATASETS

The CNN / Daily Mail dataset has a considerably bigger amount of instances than the other 2 datasets.

The following table shows the differences in the average length (in characters) of the full text and the reference summary for all the datasets:

	Avg character length full text	Avg character length summary
News-summary dataset	357	57
CNN / Daily Mail dataset	4,027	297
Multi-news dataset	11,044	1,299

Table 5.2. COMPARISON OF THE LENGTH OF THE DATASETS

It is possible to observe that the News-summary dataset has small stories while the CNN / Daily Mail dataset has medium size stories, finally the Multi-news dataset has long stories.

5.2. Methods

In this section, we describe the methods used in this project to address the task of text summarization. In particular, we implement the following methods: choosing the first k sentences, word frequency-driven approach, Text Rank, Latent Semantic Analysis (LSA) and PreSumm [40].

5.2.1. Choosing the first k sentences

It uses a simple idea. News articles generally say the most important information at the beginning of the article. An easy summary can be made by just taking the first k sentences, where 'k' is decided by using the validation datasets.

5.2.2. Word frequency-driven approach

This approach uses the word frequency to rank and select the most relevant sentences in a text. The implementation is based in [43]. It can be performed by the following steps:

- Preprocessing of the data: Remove null data and stop words.
- Create a dictionary of the words and their frequencies in the text.
- Divide the text in sentences.
- Remove the stop words.

- Score sentences with Term frequency (TF).
- Calculate the average scores of the sentences. This is considered the threshold score for selecting the sentences.
- Generate summary with sentences with scores above the threshold.

In this approach, the only hyper-parameter is the threshold. The validation dataset of each dataset was used to determine the best threshold by tuning slightly up or down.

5.2.3. TextRank

TextRank[44] is a graph-based ranking model which is an adaptation of the PageRank [31] Algorithm for NLP. PageRank algorithm gives a score between 0 and 1 to each node, the sum of the score of all the nodes is always 1. The score can be interpreted as the probability of being in a node by moving randomly across the graph. The nodes with bigger scores are the most important ones in the graph. PageRank scores can be calculated following the next steps:

1. Each node in the graph starts with the same score. The score is 1 divided by the number of nodes. This way it is guaranteed that the sum of the score of the nodes is 1.
2. Then Each node divides its score by the number of links that have. This divided score is transferred to each of the neighbors. The new score of each node is the sum of the revived score from the neighbors nodes.
3. Repeat the previous step until it converges (the scores do not change) or the difference in scores falls below a certain predefined threshold or a maximum number of iterations is reached.

Example of how PageRank is calculated with 3 iterations for the given graph:

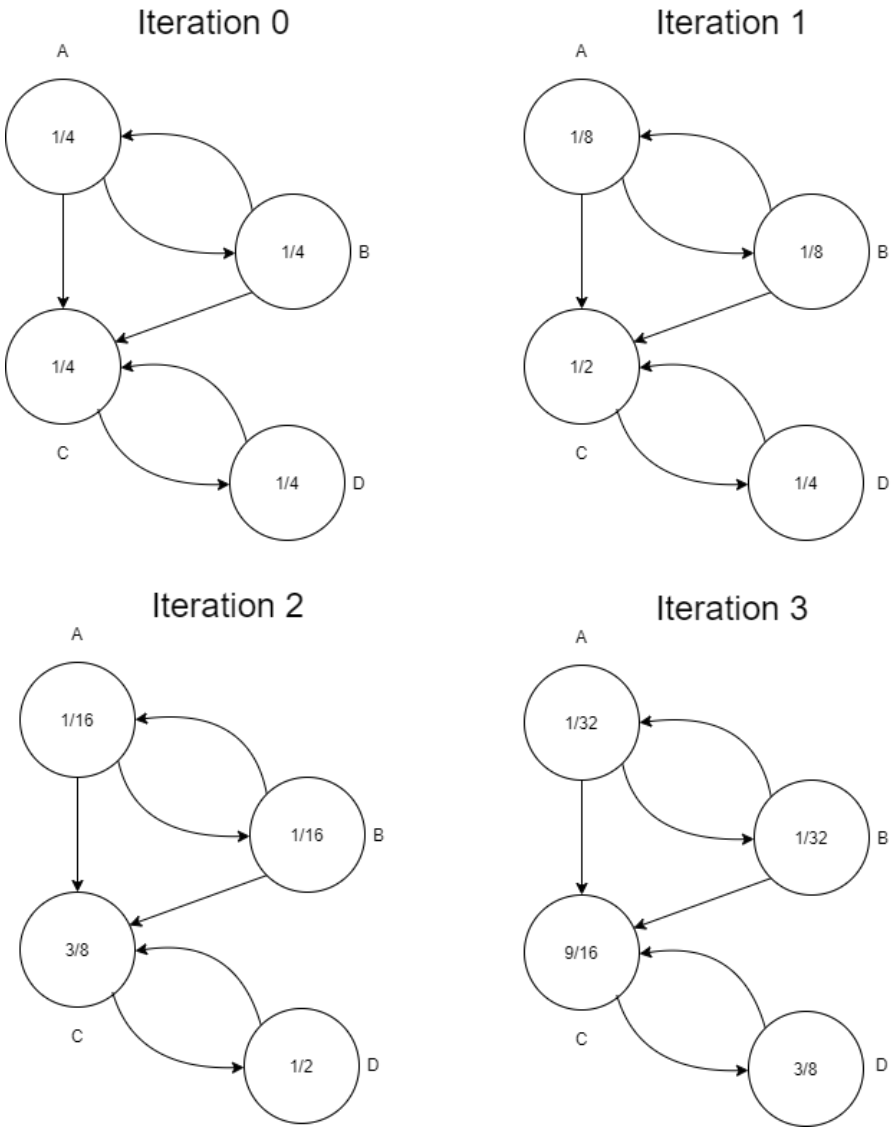


Fig. 5.1. Page rank example

Since we are doing extractive text summarization, each sentence is a node and the number of shared words the sentences are the links. For the implementation the library pyTLDR for python was used. The summary is performed by taking the nodes with the biggest scores. The algorithm requires a parameter that determines the number of sentences that the summary will have.

5.2.4. Latent Semantic Analysis (LSA)

"Latent Semantic Analysis is an algebraic-statistical method that extracts hidden semantic structures of words and sentences" [32] . It is an unsupervised method that uses SVD to divide the text into several topics and then select the most important sentences. It uses the implementation of LSA in the python library pyTLDR which follows the steps described in [32]. The steps in this approach are:

1. Remove the stop-words.
2. Create the input matrix, each row is a sentence and each column is a word. In particular, it is a binary matrix, where each value is 1 if the word is in the sentence and 0 if it is not.
3. Calculate the SVD, an algebraic method that takes a matrix and decomposed into 3 matrices:

$$A = U \Sigma V^T$$

"Where A is the input matrix (m x n); U is words x extracted concepts (m x n); Σ *represents scaling values, diagonal descending matrix (n x n); and V is sentences x extracted concepts (n x n)"* [32]

4. Finally, select the important sentences using SVD.

5.2.5. PreSumm

Presumm uses a modified version of BERT[24] as an encoder. Figure 5.2 shows the differences. The left is the original BERT , the right the modified version for summarization. The first row is the input sequence (already tokenized). Both start by the special classification token [CLS]. The original BERT uses a [SEP] token to separate sentences of one document. However, this version is modified by using after [SEP] again the token [CLS]. In this way, this token collects the features of the prior sentences. [40]

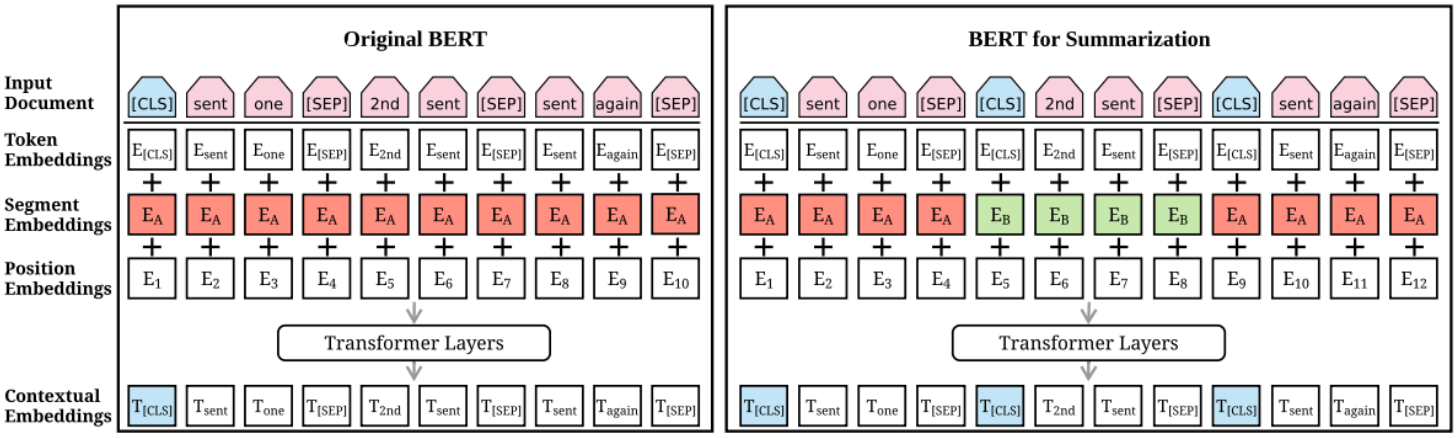


Fig. 5.2. Comparison original BERT and BERT for summarization [40]

In the extractive version of PreSumm, several inter-sentence Transformer layers are used after the BERT to learn document level features. The first transformer layers capture

features close sentences while the later layers capture the features of the document. The final layer is a sigmoid classification layer that determines if a sentence is included or not in the summary. The loss function is a binary classification entropy of prediction that is compared against the human generated summary.

For each dataset, a separate model has been trained. This method requires particular prepossessing steps. The prepossessing steps needed are described in https://github.cneedom/nlpyang/PreSumm and have been implemented in https://github.com/Andres-Escalante/Extractive-Textsummarization. These particular steps are needed due to the use of the modified version of BERT.

The implementing PreSumm has 2 possible approaches, starting with BERT or starting with a pretrained model provided by the authors that have been trained with dataset 2. For the datasets 1 and 3, it was tested with both options, no significant difference was found. The results provided in the next section were obtained using the first approach. For dataset 2 to avoid data leakage (do the test with the same data that was used during training) a new model was trained starting from BERT.

All the methods were developed in python using mostly Google Colab since it provides a free platform with access to discrete GPU, this decision causes a few issues that are discussed in section 7.1. Since the Presumm Algorithm was developed in python it was the better choice to use it. For the other methods python was also used since they are good libraries for Natural Language Processing like nlpk and pyTLDR. Finally python is the best choice to perform the evaluation since there is a library to get the rouge scores.

6. EVALUATION

This chapter shows the results of the different methods studied on the three datasets. The scores shown are F1 scores of ROUGE-1, ROUGE-2, ROUGE-L.

As baseline, we propose the original text as summary.

6.1. Dataset1: Indian times and Guardian

Method	ROUGE-1	ROUGE-2	ROUGE-L
Baseline (no summary)	0.1854	0.0724	0.1925
First k sentences	**0.3215**	**0.1332**	**0.2847**
Word Frequency	0.1868	0.0703	0.1892
TextRank	0.2586	0.0978	0.2289
LSA	0.2638	0.1002	0.2322
PreSumm	0.2348	0.1004	0.2401

Table 6.1. ROUGE F1 SCORES DATASET 1

The Hindu, Indian times and Guardian dataset has considerable shorter news (see table 5.1). The ROUGE-1 score of the baseline approach is 0.1854. All methods are able to to improve it, except the word frequency-driven approach. This may be due to the short news articles, it does not have enough words to score the sentences correctly. The best method with the highest scores in all 3 metrics is first k sentences, with k = 1. In other words, this means that just selecting the first sentence is the best way to do an extractive summary for this dataset. LSA and TextRank have a better score than PreSumm in ROUGE-1 however PreSumm have a better score in ROUGE-2 and ROUGE-L. Since in [28] the author of ROUGE concluded that the metrics ROUGE-1 and ROUGE-L performed best for headline like summaries, it can be considered that LSA is the second best method.

6.2. Dataset 2: CNN / Daily Mail dataset

Method	ROUGE-1	ROUGE-2	ROUGE-L
Baseline (No summary)	0.1372	0.0749	0.2061
First k sentences	0.3420	0.1487	0.3420
Word Frequency	0.1994	0.0875	0.2476
TextRank	0.2918	0.1022	0.2706
LSA	0.2688	0.0930	0.2624
PreSumm	**0.4102**	**0.1812**	**0.3965**

Table 6.2. ROUGE F1 SCORES DATASET 2

The CNN / Daily Mail dataset brings different results in comparison with the previous dataset. In this dataset, the news articles are longer than in the previous one (see table 5.2). This makes this task more difficult. The results are in table 6.2. As expected, the baseline scores are very poor compared to the other methods. PreSumm achieved the best results by a big margin, which can be explained because this method was created and tested in this dataset. Also it is known that deep learning algorithms work better with big amounts of data. In table 5.1, it can be seen that this dataset has a lot of samples which results in having more than double the training samples than the other datasets used in this project. The Word frequency is the worst performing method, The data is still not long enough to be able to score the sentences as good as the other methods. In this dataset TextRank outperforms LSA in all 3 metrics in contradiction with the previous dataset results. However First k sentences is the second best method with k = 4. In this dataset there is no differences in the ranking of the methods for the different metrics.

6.3. Dataset 3: Multi-news

Method	ROUGE-1	ROUGE-2	ROUGE-L
No summary (baseline)	0.2147	0.1015	0.2537
First k sentences	**0.3612**	**0.1175**	**0.2958**
Word Frequency	0.3106	0.0994	0.2781
TextRank	0.3417	0.0985	0.2695
LSA	0.3351	0.0957	0.2631
Presumm	0.3218	0.1105	0.2828

Table 6.3. ROUGE F1 SCORES DATASET 3

As it was shown in Table 5.1, the Multi-News dataset has the longest news and summaries. However it has the least instances of the 3 datasets (Table 5.2) The baseline approach provides better results over the previous datasets. This is caused by how the summaries have been performed. In this dataset it is a fact that the editor of the article made the summary meaning that the sentences in both texts are very similar, this derives in higher baseline scores. Moreover, all methods proposed overcome the baseline approach. The approach based on choosing the first K sentences (with k=12) has the highest scores in all 3 metrics. The very long sequences probably affect Presumm, which was not able to select the sentences with as good results as in the CNN / Daily Mail dataset. Also PreSumm was penalized by the fewer instances of this dataset, which resulted in less data to train the model. All methods have a score over 0.3. It can be concluded that the longer text and summary made the methods obtain better results. Word frequency benefited the most of the long sentences, it has a much higher score in this dataset compared to the rest datasets. It can be explained due to the longer text means more vocabulary and sentences to examine the frequency.

7. CONCLUSIONS

The goal of this project is to study the most popular methods for extractive text summarization. We have used three different datasets to evaluate these methods. Fist k sentences prove to be the best overall method being the best performing in dataset 1 and 3, and the second best in dataset 2. First k sentences is the best method for the approaches that do not require training outperforming Word frequency, LSA and TextRank in all datasets. PreSumm was the best performing method in dataset 2 being the only method across all dataset that managed to obtain a F1 ROUGE-1 superior to 0.4. However it needs a lot of data (instances) with a known solution to be trained, it also takes the most amount of time to create a model.

For future work it would be interesting to see how abstractive methods perform in these 3 datasets and compare them to the extractive methods. Also the use of other datasets that are not related to the news domain would be interesting. Other languages other than English could be a topic for future research by either selecting new datasets with other languages or translating the ones used. It would be interesting to see if there is an impact in performance of these methods for different languages.

ROUGE is a widely adopted method for evaluation, however the evaluation of the performance in text summarization is still an open topic for research where new methods can help to better evaluate the models being developed.

Text summarization is always oriented to improve the life quality of people. Therefore, there is still room to keep improving and create tools which make it easier to use them for the final users.

7.1. Problems and challenges overcame

Google Colab is a great platform which provides fast and easy access to a lot of resources and virtual machines ready to code and train machine learning models. However, these free resources are limited. There is a limit to access to a discrete GPU of 12 hours per day and it is not guaranteed. Although this time is reasonable, training machine learning models could take much more time. For training, testing and optimizing the PreSumm algorithm .it was a process that took many hours and and test runs, which were slowed down by this limitation.

The datasets used, specially the Daily CNN dataset, had a lot of instances. However, Google drive struggled to have more than 2000 files in one directory even though each file was only a few kilobytes. This problem is due to its file manager system. For example, obtaining a list of the files of a directory usually causes a timeout and consequently the task fails. This is a known limitation. Google suggests to use a lot of directories. To solve

this issue, the best solution was to work in a local machine for the few steps where this problem is encountered. Moreover, sometimes Google Drive takes hours to update and show the saved models. These limitations are very understandable since Google Colab and Google Drive are free services but it has to be taken into account.

Acknowledgement

This work was supported by the Research Program of the Ministry of Economy and Competitiveness - Government of Spain, (DeepEMR project TIN2017-87548-C2-1-R) and the Interdisciplinary Projects Program for Young Researchers at Universidad Carlos III of Madrid founded by the Community of Madrid (NLP4Rare-CM-UC3M).

8. TIMELINE

This section describes a timeline of the work done for the thesis.

Task	Start date	End date
1. Selection of the topic	10 February	17 February
2. State of the art investigation	18 February	30 April
3. Specify objective of the thesis	18 February	25 February
4. Investigation of text summarization methods	25 February	15 march
5. Investigation of evaluation methods	16 March	29 March
6. Implementing Deep learning method dataset 1	30 March	26 April
7. Implementing non deep learning methods	27 April	3 May
8. Evaluation of the methods	4 May	15 May
9. Conclusion analysis	16 May	28 May
10. Change in the objective of the TFM	29 May	7 June
10. Selection of dataset 2	8 June	10 June
11. Implementing Deep learning method for dataset 2	10 June	26 June
12. Implementing non deep learning methods (dataset2)	29 June	5 July
13. Evaluation and conclusion extraction	5 July	12 July
14. Selection of dataset 3	12 July	13 July
15. Implementing Deep learning method dataset 3	13 July	20 July
16. Implementing non deep learning methods (dataset3)	20 July	26 July
17. Evaluation and final conclusion extraction	26 July	31 July

Table 8.1. TIMELINE

From 1 to 9, I still was taking courses from the master. In early June, I had a few exams that were delayed due to COVID-19. This explains why some tasks took time to make. At 9, I looked at the results and they were not satisfactory. I decided to redo the thesis with more datasets. Finally, I decided to add the third and final dataset.

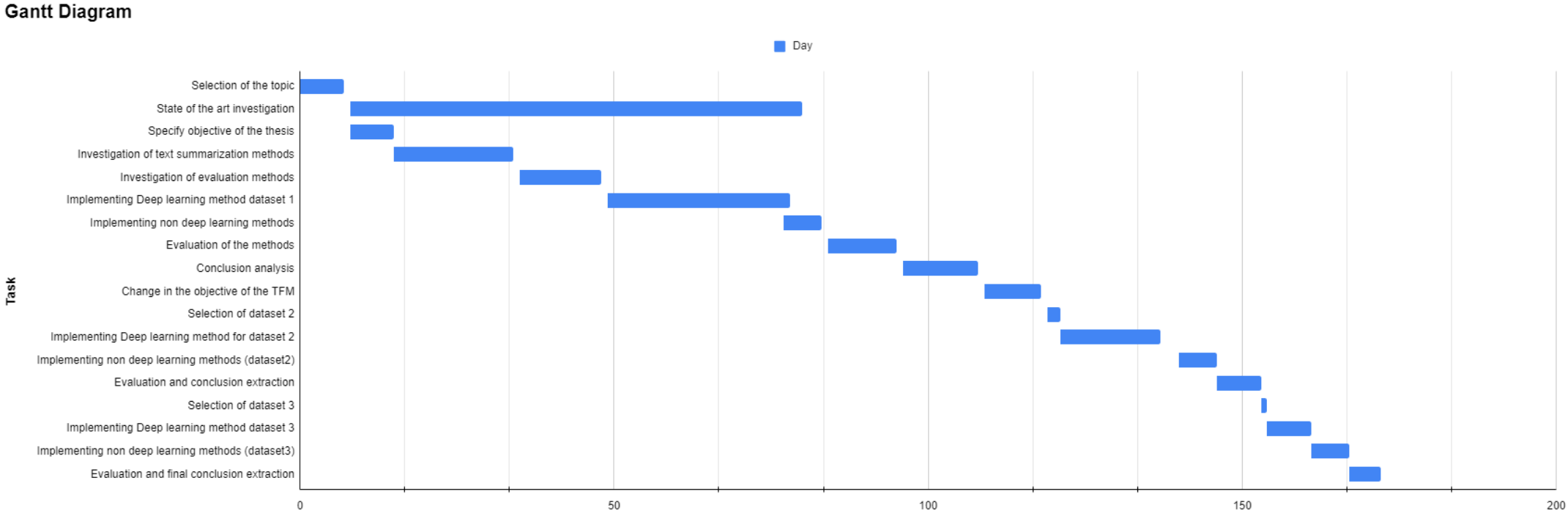


Fig. 8.1. Gantt Diagram

BIBLIOGRAPHY

[1] P. Modaresi, P. Gross, S. Sefidrodi, M. Eckholf, and S. Conrad, "On (commercial) benefits of automatic text summarization systems in the news domain: A case of media monitoring and media response analysis", *CoRR*, vol. abs/1701.00728, 2017. arXiv: 1701.00728. [Online]. Available: http://arxiv.org/abs/1701.00728.

[2] A. Nenkova and K. McKeown, *Automatic Summarization*. Jun. 2011, vol. 5, pp. 103–233. DOI: 10.1561/1500000015.

[3] M. Allahyari *et al.*, *Text summarization techniques: A brief survey*, 2017. arXiv: 1707.02268 [cs.CL].

[4] E. Cambria and B. White, "Jumping nlp curves: A review of natural language processing research [review article]", *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014.

[5] D. M. J. Garbade. (2018). A simple introduction to natural language processing, [Online]. Available: https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32#:~:text=Natural%5C%20Language%5C%20Processing%5C%2C%5C%20usually%5C%20shortened,a%5C%20manner%5C%20that%5C%20is%5C%20valuable. (visited on 09/30/2010).

[6] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]", *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, 2013. arXiv: 1310.4546 [cs.CL].

[8] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://www.aclweb.org/anthology/D14-1162.

[9] E. Alpaydin, *Introduction to Machine Learning*, 2nd. The MIT Press, 2010.

[10] L. E. Peterson, "K-nearest neighbor", *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009, revision #137311. DOI: 10.4249/scholarpedia.1883.

[11] D. W. H. Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. John Wiley and Sons, Apr. 2013.

[12] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. WORLD SCIENTIFIC, 2002. DOI: 10.1142/5089. eprint: https://www.worldscientific.com/doi/pdf/10.1142/5089. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/5089.

[13] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[14] S. Jadon, "Introduction to different activation functions for deep learning", *Medium, Augmenting Humanity*, vol. 16, 2018.

[15] Escachator. (Jan. 2019). How does gradient descent and backpropagation work together?, [Online]. Available: https://datascience.stackexchange.com/questions/44703/how-does-gradient-descent-and-backpropagation-work-together#:~:text=Back%5C%2Dpropagation%5C%20is%5C%20the%5C%20process,down%5C%20through%5C%20the%5C%20loss%5C%20function. (visited on 08/03/2020).

[16] P. Marius, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks", *WSEAS Transactions on Circuits and Systems*, vol. 8, Jul. 2009.

[17] C. Thomas, "An introduction to convolutional neural networks", *Towars data science*, May 2019.

[18] M. Venkatachalam, "Recurrent neural networks", *Towars data science*, Mar. 2019.

[19] K. Cho *et al.*, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: 1406.1078 [cs.CL].

[20] S. Kostadinov, "Understanding encoder-decoder sequence to sequence model", *Towards Data Science*, Feb. 2019. [Online]. Available: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[22] M. Phi, *Illustrated guide to lstm's and gru's: A step by step explanation*, 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (visited on 08/03/2020).

[23] A. Vaswani *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. arXiv: 1810.04805 [cs.CL].

[25] Y. Zhu *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books", in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[26] (2019). MODELOS DE EXÁMENES curso 2018/2019, lengua castellana y literatura ii, [Online]. Available: https://www.uc3m.es/pruebasacceso/modelos-examenes (visited on 07/07/2020).

[27] S. Bittrich *et al.*, "Application of an interpretable classification model on early folding residues during protein folding", *BioData Mining*, vol. 12, Jan. 2019. DOI: 10.1186/s13040-018-0188-2.

[28] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries", in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013.

[29] R. A. García Hernández and Y. Ledeneva, "Word sequence models for single text summarization", Mar. 2009, pp. 44–48. DOI: 10.1109/ACHI.2009.58.

[30] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment", *J. ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999. DOI: 10.1145/324133.324140. [Online]. Available: https://doi.org/10.1145/324133.324140.

[31] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.", Stanford InfoLab, Technical Report 1999-66, Nov. 1999, Previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/.

[32] M. Ozsoy, F. Alpaslan, and I. Cicekli, "Text summarization using latent semantic analysis", *J. Information Science*, vol. 37, pp. 405–417, Aug. 2011. DOI: 10.1177/0165551511408848.

[33] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks", *CoRR*, vol. abs/1704.04368, 2017. arXiv: 1704.04368. [Online]. Available: http://arxiv.org/abs/1704.04368.

[34] K. M. Hermann *et al.*, "Teaching machines to read and comprehend", in *Advances in neural information processing systems*, 2015, pp. 1693–1701.

[35] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning", in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1747–1759. DOI: 10.18653/v1/N18-1158. [Online]. Available: https://www.aclweb.org/anthology/N18-1158.

[36] Y. Dong, Y. Shen, E. Crawford, H. van Hoof, and J. C. K. Cheung, "BanditSum: Extractive summarization as a contextual bandit", in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3739–3748. DOI: 10.18653/v1/D18-1409. [Online]. Available: https://www.aclweb.org/anthology/D18-1409.

[37] Q. Zhou *et al.*, "Neural document summarization by jointly learning to score and select sentences", in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 654–663. DOI: 10.18653/v1/P18-1061. [Online]. Available: https://www.aclweb.org/anthology/P18-1061.

[38] M. Zhong, P. Liu, D. Wang, X. Qiu, and X. Huang, *Searching for effective neural extractive summarization: What works and what's next*, 2019. arXiv: 1907.03491 [cs.CL].

[39] O. Vinyals, M. Fortunato, and N. Jaitly, *Pointer networks*, 2015. arXiv: 1506.03134 [stat.ML].

[40] Y. Liu and M. Lapata, *Text summarization with pretrained encoders*, 2019. arXiv: 1908.08345 [cs.CL].

[41] M. Zhong *et al.*, *Extractive summarization as text matching*, 2020. arXiv: 2004.08795 [cs.CL].

[42] A. R. Fabbri, I. Li, T. She, S. Li, and D. R. Radev, *Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model*, 2019. arXiv: 1906.01749 [cs.CL].

[43] A. Panchal., *Text-summarization*, https://github.com/akashp1712/nlp-akash/blob/master/text-summarization/Word_Frequency_Summarization.py, 2019.

[44] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text", in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 404–411. [Online]. Available: https://www.aclweb.org/anthology/W04-3252.

Appendices

A. SUMMARY EXAMPLE

In this section it is going to be shown one example for each method of summaries and its corresponding scores. The example is from dataset 2 [33] [34] since it is more widely used across different papers for text summarization.

A.1. Original text

The Force is strong in Turkey – or at least it appears to be at one university where thousands of students are petitioning for a Jedi temple to be built on campus. The petition, which was started by a student at Dokuz Eylul University in the western province of Izmir, so far has almost 5,500 signatures. There are less and less Jedi left on the Earth, the petition says. It adds that uneducated Padawan – the novice Jedis in George Lucas Star Wars film franchise – are moving to the dark side. ... To find the balance in the Force, we want a Jedi temple. The page on Change.org also features a still of Jedi Grand Master Yoda from Star Wars: Episode II – Attack of the Clones teaching young Jedis how to use a light saber. I am signing because the nearest temple (is) billions of light years away, one supporter wrote on the petition page. Another supporter wrote: We want freedom of worship. There are mosques everywhere, but no Jedi temple! The petition was started by Akin Cagatay Caliskan, an 18-year-old computer science student from Ankara. Caliskan says he is surprised by the impact his petition has made: I did not expect so many supporters. I thought maybe it might (have) 100. The satirical document does have a serious side. An online debate started last month when Mehmet Karaca, the rector of Istanbul Technical University, said he would build a mosque on campus if there was enough support. Karaca was referencing another petition asking a mosque be built on campus, which almost 200,000 students signed. Twenty thousand students at ITU also signed a separate petition in response, demanding a Buddhist temple on campus. I cannot fulfill my religious needs and cannot afford to go to the nearest Buddhist temple 2,000 miles away, Utku Gurcag Boratac from Istanbul wrote as her reason for signing. I want to fulfill my religious needs on campus, said another. This is just the latest flurry in a long-standing discussion around civil and religious freedoms in the secular republic. Turkeys secularist opposition has accused Turkish President Recep Tayyip Erdogan and his party, the Islamist-based Justice and Development Party, of trying to instill conservative Islamic values into everyday life. Late last year, the head of Turkeys Islamic religious affairs authority, the Diyanet, announced plans to build 80 mosques on university campuses across the country. The Diyanet sparked discussion a few months earlier by opening a mosque on the campus of Erzurum Ataturk University in the eastern province of Erzurum, according to Turkish daily newspaper Hurriyet.

A.2. Human generated summary

5,500 signatures call for Jedi temple after similar petition asked for campus mosque . Petition started by a student at Dokuz Eylul University in the western province of Izmir .

A.3. First k sentences (k=4)

The Force is strong in Turkey – or at least it appears to be at one university where thousands of students are petitioning for a Jedi temple to be built on campus. The petition, which was started by a student at Dokuz Eylul University in the western province of Izmir, so far has almost 5,500 signatures. There are less and less Jedi left on the Earth, the petition says. It adds that uneducated Padawan – the novice Jedis in George Lucas Star Wars film franchise – are moving to the dark side.

A.4. Word Frequency

The petition, which was started by a student at Dokuz Eylul University in the western province of Izmir, so far has almost 5,500 signatures. There are less and less Jedi left on the Earth, the petition says. ... To find the balance in the Force, we want a Jedi temple. I am signing because the nearest temple (is) billions of light years away, one supporter wrote on the petition page. Another supporter wrote: We want freedom of worship. There are mosques everywhere, but no Jedi temple! The petition was started by Akin Cagatay Caliskan, an 18-year-old computer science student from Ankara. I thought maybe it might (have) 100. The satirical document does have a serious side. Karaca was referencing another petition asking a mosque be built on campus, which almost 200,000 students signed. Twenty thousand students at ITU also signed a separate petition in response, demanding a Buddhist temple on campus. I want to fulfill my religious needs on campus, said another.

A.5. Text Rank

The Force is strong in Turkey – or at least it appears to be at one university where thousands of students are petitioning for a Jedi temple to be built on campus.Late last year, the head of Turkeys Islamic religious affairs authority, the Diyanet, announced plans to build 80 mosques on university campuses across the country.

A.6. LSA

The Force is strong in Turkey – or at least it appears to be at one university where thousands of students are petitioning for a Jedi temple to be built on campus.Twenty thousand

students at ITU also signed a separate petition in response, demanding a Buddhist temple on campus.Late last year, the head of Turkeys Islamic religious affairs authority, the Diyanet, announced plans to build 80 mosques on university campuses across the country.

A.7. PreSumm

the petition , which was started by a student at dokuz eylul university in the western province of izmir , so far has almost 5,500 signatures . the force is strong in turkey – or at least it appears to be at one university where thousands of students are petitioning for a jedi temple to be built on campus ." there are less and less jedi left on the earth , " the petition says .

A.8. Scores of the examples

Method	Rouge-1	Rouge-2	Rouge-L
No summary	0.0958	0.0612	0.1473
First k sentences	0.3333	0.2372	0.4040
Word Frequency	0.2105	0.1489	0.2857
Text Rank	0.2117	0.0240	0.1866
LSA	0.1941	0.0198	0.2117
PreSumm	0.4299	0.2857	0.5116

Table A.1. SCORES OF THE EXAMPLE