Artificial intelligence deep learning text summarization and sentiment analysis tool

by
Danylo Fedorov

Engineering Project Report

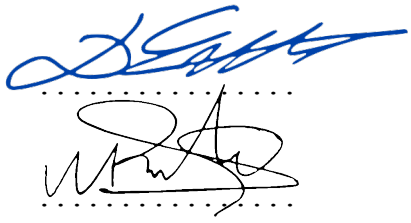
Yeditepe University
Faculty of Engineering
Department of Computer Engineering
2023

Artificial intelligence deep learning text summarization and sentiment analysis tool

APPROVED BY:

Prof. Dr. Emin Erkan Korkmaz
(Supervisor)

Assist. Prof. Dr. Dionysis Goularas .....



Assistant Prof. Dr. Mustafa Mutluoğlu .....

DATE OF APPROVAL: 17/06/2023

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Emin Erkan Korkmaz for his guidance and support throughout my project.

Also I would like to thank my parents for their support and encouragement throughout my education up to the present.

ABSTRACT

Artificial intelligence deep learning text summarization and sentiment analysis tool

Deep learning artificial intelligence text summarizer is the technique of reducing a given text to a shorter version, which can be performed in two ways: an extractive or an abstractive summarization.Extractive summarization, which can extract vital phrases and terms from the given text. The underlying the concept is to create a summary with the aid of selecting the maximum number of essential phrases from the given text.

But on the other hand, there is Abstractive Summarization which involves the generation of totally new terms that seize the meaning of the input sentence. The underlying concept is to position a strong emphasis on the shape — aiming to generate a grammatical precis thereby requiring advanced language modeling techniques. Both ways are used by humans in daily life.

In addition to the use of the summarization technique, the sentiment analysis will be used and will be performed in two ways: by determining the polarity(Irrelevant, Negative, Positive, Neutral) and relevance(World, Sport, Business, Sci/Tech) of the text. By analyzing the sentiment of the text, we can achieve a better understanding of the emotions and opinions expressed within the given content.

Some problems aspects of human life can be solved by deep learning artificial intelligence, which can make life a little bit easier. This paper, will be focused on the better way for text summarization via deep learning artificial intelligence, a deep dive into this topic and all aspects that are required to implement it. In conclusion, will be explained the methodologies that were used and what was achieved, and how it can be improved.

ÖZET

Artificial intelligence deep learning text summarization and sentiment analysis tool

Deep learning artificial intelligence text summarization techniques aim to condense text into shorter versions. There are two approaches to achieving a goal: extractive and abstractive summarization. Extractive summarization selects vital phrases from the text, while abstractive summarization generates new terms capturing the input's meaning. Both methods are used by humans. Sentiment analysis, used alongside summarization, determines polarity (irrelevant, negative, positive, neutral) and relevance (world, sport, business, sci/tech) to understand emotions and e-content of the text. Deep learning artificial intelligence can address human challenges, simplifying life. This paper focuses on implementing text summarization via deep learning artificial intelligence, exploring methodologies, achievements, and areas for improvement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

NLP	Natural Language Processing
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
AI	Artificial Intelligence
CUDA	Compute Unified Device Architecture
GUI	Graphical User Interface
GPU	Graphics processing unit
CPU	Graphical User Interface
T5	Text-to-Text Transfer Transformer
BARTScore	BART Scoring Metric
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BART	Bidirectional Auto-Regressive Transformers

1. INTRODUCTION

1.1. Deep learning

Deep learning evolved from biological neural networks found in the brain. The idea behind it is machine learning and artificial intelligence (AI) which is inspired by the structure and function of the human brain called. Deep learning involves teaching computers to learn. with examples, mimicking the way people naturally learn. The recent increase in focus on deep learning is deserved, as it has had unprecedented results. By learning directly from images, text, or audio, deep learning models can achieve remarkable accuracy, surpassing human-level performance in some cases. These models are trained using heavily labeled datasets and multi-layer neural network architectures.

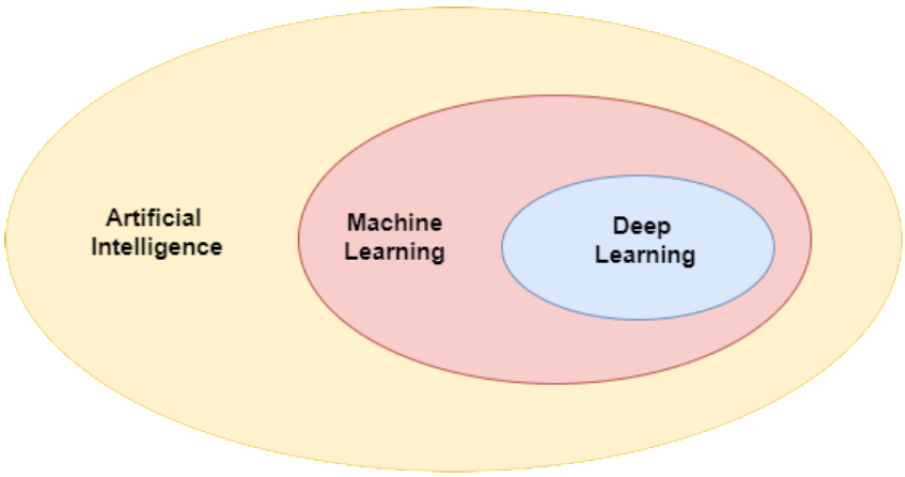


Figure 1.1. Deep learning structure

1.1.1. Artificial neural network

Artificial neural networks (ANNs) are inspired by the functioning of the unique in its kind human brain. The human brain comprises around 86 billion neurons, interconnected by axons. Neurons receive stimuli from the external environment or sensory organs through dendrites. These inputs generate electric impulses that rapidly propagate through the neural network. Neurons can transmit messages to other neurons to address specific tasks or choose not to forward them.ANNs consist of multiple nodes that simulate the behavior of biological neurons in the human brain. These neurons are connected through links, allowing them to

interact with one another. The nodes receive input data and perform basic operations on it. The output of these operations, referred to as the activation or node value, is then passed to other neurons. Each link in the network is associated with a weight value. ANNs can learn, achieved by adjusting the weight values.

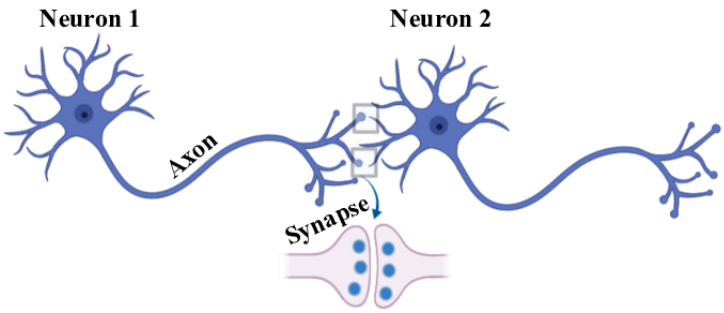


Figure 1.2. Human Neurons

1.1.2. Convolutional neural network

Convolutional networks[1] originate from biological processes in the visual cortex of the human brain, which contains specialized cells called neurons that respond to specific visual stimuli.The concept of CNNs was first introduced in the 1980s by Yann LeCun and other researchers Layers are used that apply filters or kernels to the input data, enabling the network to automatically learn the spatial hierarchy of patterns and features. These transformation layers are followed by pooling layers, which reduce the dimensionality of the feature maps, and the fully connected layers that perform the final classification or regression operations.

1.1.3. Neural Networks for Natural Language Processing

Natural language processing (NLP) neural networks refer to a type of neural network specifically designed to process and analyze natural language data. NLP neurons harness the power of deep learning algorithms to understand, interpret, and acquire human speech. The machinery for these websites has revolutionized NLP by enabling text understanding and generation, language translation, sentiment analysis, data aggregation, and a variety of other language-related functions. The roots of the neural network of NLP can be traced back to the early days of artificial intelligence and neural networks in the 1940s and 1950s but the advent of deep learning and the availability of large datasets over the century this 21st edition spurred major advances in NLP. In 2017, the Transformer architecture was introduced, which revolutionized NLP tasks. The Transformer model leveraged self-attention mecha-

nisms to capture contextual relationships between words in parallel, eliminating the need for recurrent connections. Transformers became the cornerstone of many state-of-the-art NLP models, including BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). The success of NLP neural networks can be attributed to their ability to learn complex representations from large amounts of text data, effectively capturing semantic and syntactic patterns in language. Their applications have expanded to various domains, including chatbots, virtual assistants, sentiment analysis tools, and language translation systems, among others.

1.2. Deep learning Text summarization

The modern life of humans requires fast access to good and quality sources of information that will be used to achieve personal requirements in any field people face it. It is important to have information but it is always better to have a summarized version of it with which humans can communicate and organize their time spending. Text summarization is one of the possible solutions, it can be performed over traditional methods but I strongly believe that the deep learning artificial intelligence approach is the best possible way to such a problem. And there are reasons why: Deep learning approach is learning directly from a large amount of data which is usually located in the datasets, human-readable datasets are designed to help to interact with data more comfortably and make training of it easier. By training the model on a big dataset using a tokenizer, we can capture complex patterns and relationships that might be difficult to define over traditional ways such as handcrafted approaches. The tokenizer helps in breaking down the text into meaningful units which is machine-readable, allowing the model to understand the structure and context more effectively. The deep learning approach allows us to generalize better and adapt to different types of texts.

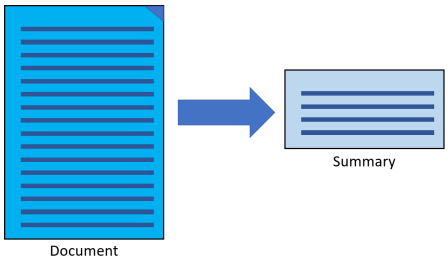


Figure 1.3. Text summarization

There are mainly exist two different approaches. The text summarization problem can be solved by implementing the Abstractive and Extractive way of summarization. Let us discuss

both approaches in more detail to get a clear understanding of both approaches:

1.2.1. Abstractive summarization

Abstractive summarization[2] [3] generates summaries by understanding the direct meaning of the given text and generating new phrases or sentences in a more human-like manner. Basically, it is performed by understanding the source text and generating the summary by saving the meaning of the original text. However, abstractive summarization poses a challenge that requires a deep understanding of the source text and the ability to generate language that is both grammatically correct and semantically meaningful in case of generating coherent and contextually appropriate sentences. Deep learning and Neural networks are designed to solve these challenges by learning from large amounts of data and trained to generate summaries that capture the essential meaning of the text.Such approach aims to reduce this issue by generating summaries that are not limited to the exact sentences or phrases found in the source text, allowing more flexibility and creativity from the abstractive summarization model.

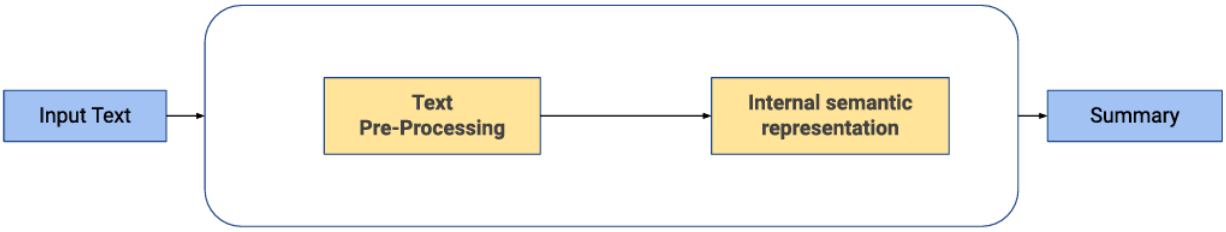


Figure 1.4. Abstractive Architecture

1.2.2. Extractive summarization

In recent years, deep learning-based text summarization techniques, leveraging the power of transformers has gained significant attention. Transformers are a powerful neural network architecture that has demonstrated exceptional performance in various natural language processing (NLP) tasks. They use self-attention mechanisms to capture relationships between words or tokens in a sequence, enabling a deep understanding of the context.

When it comes to Extractive summarization [4] [3] which aims to select and assemble the most significant sentences or phases from the source text, transformers have been employed

as a crucial component of the summarization pipeline. This approach does not generate a new sentence but rather extracts the most relevant information from the given text. This approach mainly focused on keeping the original structure of the text. The pipeline typically involves several steps, including text preprocessing, tokenization (performed by a tokenizer), encoding the text with a transformer-based model, and then selecting the most important sentences or phrases based on the encoded representation. Transformers help in capturing the contextual information and semantic relationships between words, enhancing the quality of the extracted summary.

By incorporating transformers into the extractive summarization pipeline, deep learning approaches can achieve more accurate and flexible summarization compared to traditional methods. The tokenizer, which breaks down the text into meaningful units, plays a vital role in representing the text for the transformers. This combination of transformers and tokenizers empowers the model to capture the essence of the source text and extract relevant information, leading to higher-quality summaries.

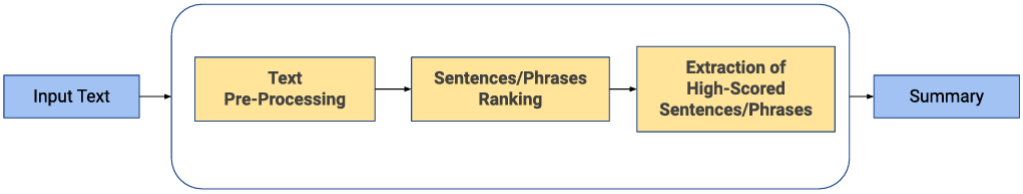


Figure 1.5. Extractive Architecture

1.2.3. Sentiment analysis

In addition to text summarization, it is also essential to classify the text into a class thought sentiment analysis Deep learning models, such as those based on CNN (Convolutional Neural Network) [1] [5]approaches have proven to be effective in predicting the sentiment or emotional tone expressed in the text such as the relevance or polarity of a given text. This classification helps us gain a powerful and automated understanding of what the text is about in terms of polarity(Irrelevant, Negative, Positive, Neutral)and Relevance(World, Sports, Business, Science/Technology).

Figure 1.6. Sentiment analysis

1.3. Terms

- *Tokenizer* is a tool that breaks down text into smaller units called tokens. These tokens can be used as input to a deep learning model. It is tokenizing the raw human-readable text into a suitable machine-readable format for input into a deep learning model. Tokenizers perform important tasks by splitting the text into individual tokens and converting words to corresponding numerical representations.

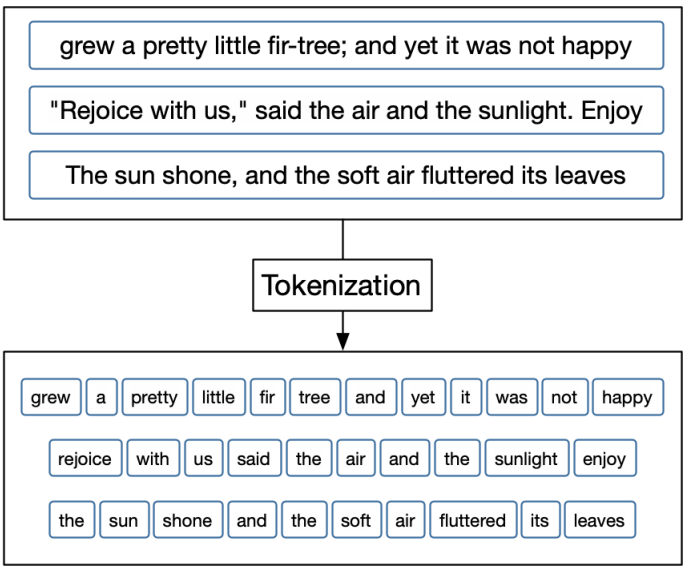


Figure 1.7. Tokenization process

- *Dataset* is a collection of text that is used to train deep learning models. Text is stored in a variety of formats which are labeled. That means each piece of the text has a

corresponding label that indicates the desired output of the model. The dataset is typically divided into two main subsets: the training and test set. Training sets are used to train the model by presenting it with corresponding data and labels to each other, it is allowing the model to learn patterns and relationships in the data. On the other hand test set is used to evaluate the training model's performance by measuring how well it generalizes examples

- A *Transformers* is the type of neural network architecture which is used for natural language processing(NLP) tasks. Transformers are well-suited for the task that requires an understanding of the text context, such as text summarization.

- *Pipelines* using transformers to perform NLP tasks. A pipeline consists of a sequence of transformers that are used to perform different steps of an NLP task. These steps typically include text preprocessing, tokenization, encoding, and selection of important sentences or phrases. The use of pipelines allows for a systematic and organized approach to text summarization.

- *Convolutional neural networks (CNNs)* are a type of deep learning model that is appropriate for extracting important features from the input text. This model identifies patterns in the input data and then uses those patterns to generate a summary.CNNs mainly used for extractive summarization which is identifying the most important sentences in a given text and then extracting those sentences to be able to create a result summary.

- *Natural language processing (NLP)*[1] is also a type of deep learning model that targets the relations between computers and human language. In the field of text summarization, the NLP aproach can be used to extract the meaning of sentences by using various methods to understand and process textual data, such as tokenization, semantic analysis, and more.NLPs mainly used for the abstractive summarization method.

- *Model* In terms of deep learning, a model refers to a mathematical representation that learns patterns and relationships from a dataset or any data. Models are specifically trained to generate logical and grammatically correct summaries of longer texts. They aim to capture prominent information from a given text.

- *Deep learning* is machine learning that uses artificial neural networks to learn from a dataset or any data. Artificial neural networks are inspired by the human brain and it is making it possible to learn complex patterns from data that would be different from traditional machine learning to achieve the same performance.

- *Training model* is the process of teaching the model to identify patterns in the text and to generate summaries or identify sentiment which is done by feeding the model a large

dataset of text and labels until can accurately predict the desired output. The model's parameters are the weights and biases that control how the model learns. As soon as it is trained,it can be used for any task it was made for.

Figure 1.8. Training the model

1.4. Motivation

1.4.1. Accessibility

Nowadays in the modern and developed world, humans are often suffering from a lack of time and it is important to provide them with high-quality information. However, the most of available information is time-consuming. Therefore, the need for an accessible source of information has become crucial to this day. Text summarization and classification based on deep learning artificial intelligence aims to solve this challenge by reducing the length of the original source of information to concise and meaningful summaries. This approach has the main advantage over traditional methods of summarization is to be able to capture complex patterns and relationships present in the data. The deep learning approach allows for better generalization and adaptability to different types of texts.

1.4.2. As a Basis for Other Applications

Text summarization is a crucial component of natural language processing(NLP) which servers as the foundation for various fields. The deep learning artificial intelligence approach is a basis for other applications with the best possible solution for efficiently overcoming massive information. By exploiting the power of text summarization performed via transformers and tokenizers, we can achieve the most possible accurate summarization and classification for humans to save humans time spending.

1.5. Scope and Limitations

Scope:

- The main task of text summarization is to provide individuals with high-quality summaries which can be performed for various purposes, such as to help people quickly understand the main idea that stands behind of given text.
- Text summarization consists of two main approaches: abstractive and extractive summarization.
- Text classification is performed by the model which predicts the sentiment of the given text and classification it for polarity and relevance.

Limitations:

- The quality of the summary depends on the quality of the input text.
- Can be difficult to determine the most important sentences in the text.
- Abstractive summarization method is more challenging than extractive summarization can be, it can be difficult to generate summaries that are quality and accurate.
- The length of the input text might be challenging due to difficulties in identifying the most important information and ensuring that the summary is completely good enough.
- The complexity of the given text might be difficult for the model to understand the meanings of the most important parts of the text and to generate a high-quality summary or define the sentiment basis of the text

1.6. Problem Definition

Generation and classification for an informative summary and sentiment prediction of input text. It is difficult to get easy and quick information that people need while they interact with an unlimited flow of data, a good text summarization application aims to help people to understand the main points of a text information fast. It is also maybe become a good tool for people with disabilities.

I strongly believe that the application which will be able to classify and summarize the given text can be built by using abstractive, extractive and sentiment analysis models trained on different datasets to provide the users with quality results.

1.7. Requirements

The application should be able to:

- Take in a piece of text as input.
- Generate a summary of the text.
- Classify the text into one of several categories.
- Allow the user to choose between extractive and abstractive summarization.
- Allow the user to choose the length of the summary.
- Allow the user to copy the summary to the clipboard.
- Allow the user to clear the input and output areas.
- Allow the user to download the output as a text file.
- Allow the user to exit the application.

Necessary software:

- Google Drive to load the trained model to a folder.
- Windows 10 or 11.
- Visual Studio Code.
- Anaconda python.
- Tenserfllow.
- Pytorch.
- CUDA core platform.

Figure 1.9. GPU with CUDA cores

Necessary hardware:

- NVIDIA GPU to train and fine-tune the models.
- Any GPU or CPU, after the model is trained it is CUDA independent.

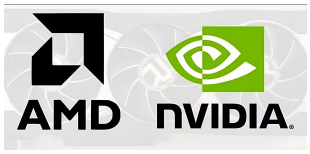


Figure 1.10. GPU CUDA independent

Tests to evaluate the project:

- Performance of the models.
- Accuracy of the summary.
- Fluency of the summary.
- Length of the summary.
- Time spent to summarize.
- Time spent to classificate.
- Sentiment of the summary and predicted class.
- User satisfaction with the application.

2. BACKGROUND

To build a text summarization and classification tool which will be able to generate a summary of abstractive, extractive approaches and to classify summary from models for polarity and relevance. It is important to make offline and online research on this problem. Text summarization and sentiment analysis are at their early stages and it can be too problematic to find a lot of resources but since this field is getting very popular these days due to the high demand for deep learning approaches to deal with big data and datasets which can be complex to do with the ordinary way of programming. We are gonna mainly focus on resources that explore CNN and NLP model architectures.

2.1. Reference website for summarization

As a reference for text summarization application, the internet source ”QuillBot Text summarizer” [6]was used to take its main features of it and put them into practice. Namely features such as:

- Choose the lenght and summarization approach of the generated summary.
- Upload the file with text.
- Count generated words.

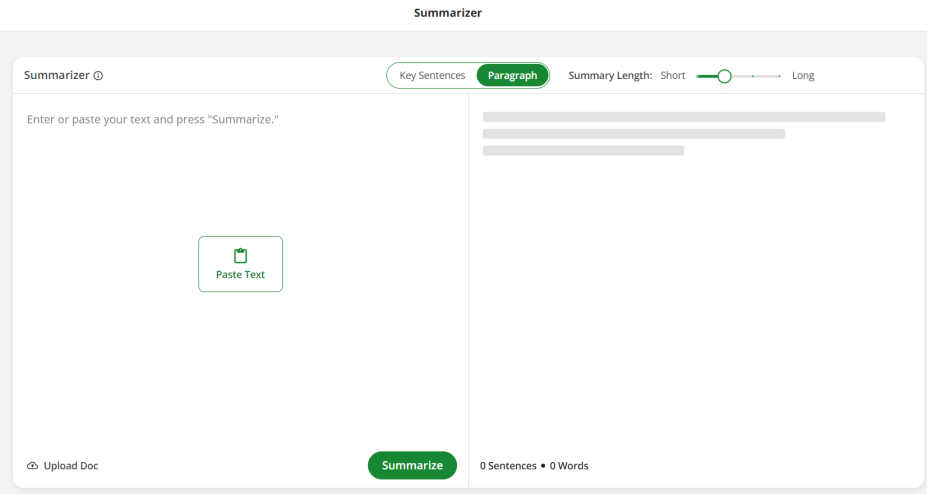


Figure 2.1. Reference for Summarization

2.2. Reference website for sentiment analysis

As an additional feature for text summarization, text classification which is based on sentiment analysis is performed and the reference internet source is ”Monkeylearn Sentiment Analysis”[7].The feature which is used in the main application are:

- Pass the text to the model
- Get the confidence of predicted class for Sentiment Analysis

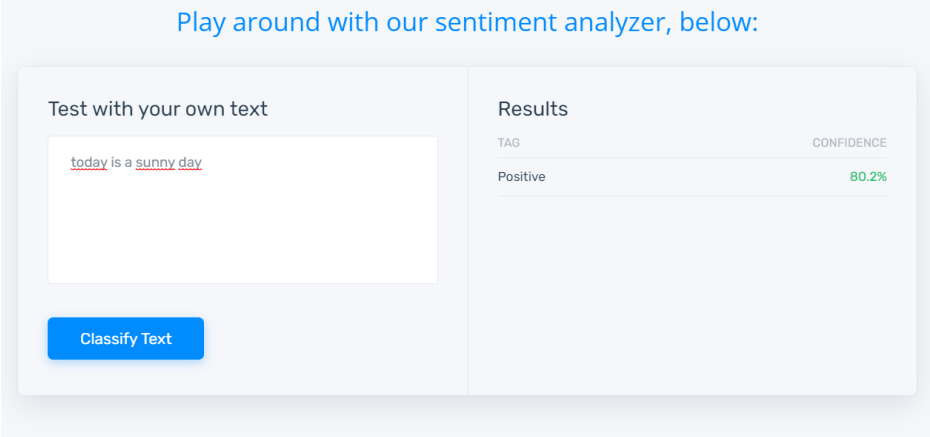


Figure 2.2. Reference for Sentiment Analysis

2.3. NLP Architecture

Vaswani et al introduced a paper ”Attention Is All You Need”, paper is states to help to understand the Transformer architecture which is adopted for various NLP tasks, including text summarization. The Transformer’s mechanism enables it to capture dependencies and generate high-quality summaries. The main features of such an approach are:

- NLP text summarization architectures summarize given text by leveraging attention mechanisms to capture global dependencies for creating a logical and Grammarly summary.
- NLP text summarization architectures can handle various languages.

2.4. CNN Architecture

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom introduced a paper ”A Convolutional Neural Network for Modelling Sentences” that analyzes the convolutional neural network architecture for modeling sentences that can be adapted for text summarization tasks. This paper introduces and explores a dynamic k-max pooling operation, which allows the model to extract the most remarkable sentences and words from input text. This work is intended to show the effectiveness of such an approach. An overall convolutional neural network is a powerful tool for various tasks, including image recognition, natural language processing, and time series analysis. The main feature of such an approach is:

- CNNs stand out at capturing local patterns and dependencies within the input data.

2.5. T5 model

For Abstractive summarization T5 model is used, it is a versatile transformer-based model that can be fine-tuned on different datasets for various NLP tasks. The Hugging Face team providing Documentation for the T5 model ”T5: Text-To-Text Transfer Transformer”. The documentation provides an overview of the T5 model architecture, explaining its structure as a sequence-to-sequence transformer model. It also provides documentation on how to fine-tune T5 for text generation tasks such as abstractive summarization by using PyTorch and TensorFlow.The main feature of such an approach is:

- T5 is a versatile model, it can perform various task.

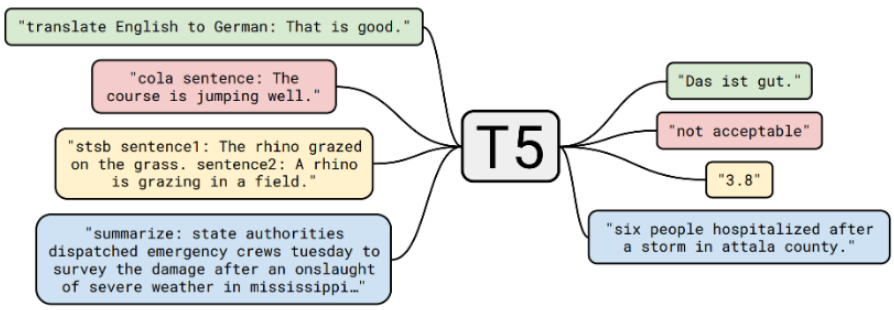


Figure 2.3. T5 model

2.6. Bart model

The Extractive summarization BART model used, it is a powerful language model that can be used for extractive summarization. The Hugging Face team provides Documentation for the BART model ”Text Generation with BART”.The Hugging Face Transformers library quality explains the BART model and how to use it for text summarization tasks. It includes code examples and guidelines on how to use BART with pipelines and transformers. The main features of such an approach are:

- BART is a relatively easy model to use.
- Pipelines provide a simplified API for text summarization.

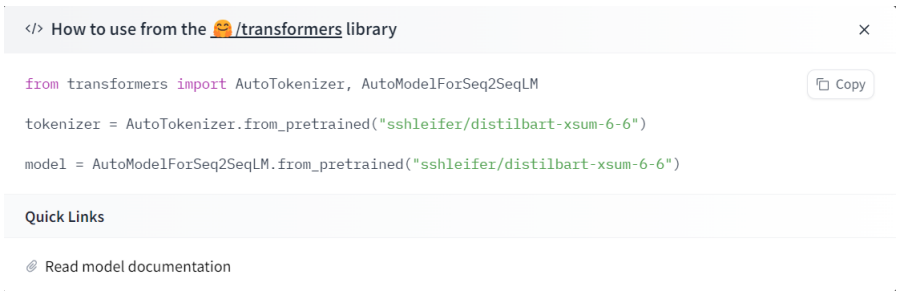


Figure 2.4. BART model

2.6.1. Pipeline and Transformers

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer introduced a research paper ”BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”, paper is stated to provide with a deep explanation of BART model architecture and its performance on different NLP tasks. The main feature of such an approach are:

- Pipeline and Transformers offer a flexible and extensible approach to text summarization.

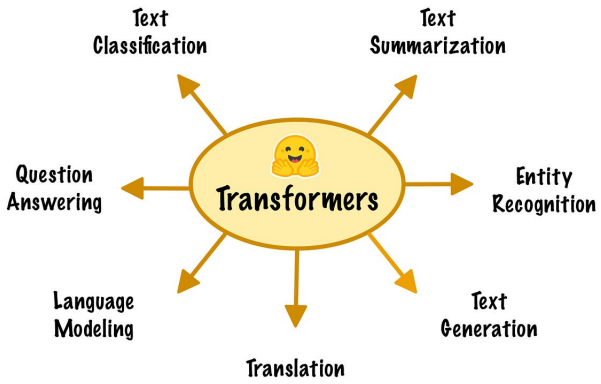


Figure 2.5. Transformers

2.7. Sentiment Analysis model

Yoon Kim introduced a seminal paper ”Convolutional Neural Networks for Sentence Classification” that proposes a convolutional neural network architecture designed for sentence classification tasks, including sentiment analysis. This paper introduces a simple CNN architecture for sentence classification. It utilizes one-dimensional convolutions to capture local patterns from the input sentences. The main feature of such an approach is:

- CNN Sentiment Analysis models are more accurate than traditional methods.

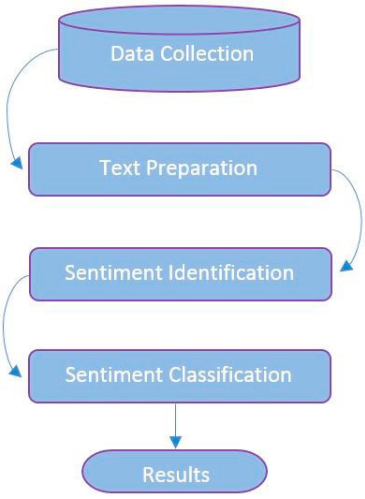


Figure 2.6. Sentiment model

3. ANALYSIS

The main goal of the project is to exploit an automatic text summarization and classification tool which is trained on various datasets.

The following steps is performed to achieve the objectives set:

- Step 1: Decide which architecture will be used for each implemented model.
- Step 2: Choose and preprocess the dataset by cleaning and splitting it.
- Step 3: Build the fine-tuned Abstractive text summarization model based on the NLP architecture T5 model.
- Step 4: Use pre-trained Extractive text summarization model based on NLP architecture BART model.
- Step 5: Build Sentiment Analysis model for polarity and relevance classification based on CNN architecture.
- Step 6: Combine all models into one application. Sentiment analysis will be performed in any scenario of user experience.

3.1. Model's Architecture

Before starting construct models it is crucial to select an appropriate model architecture for each approach of text summarization and classification. The main architectures which were chosen and used though all project work is CNN and NLP.

CNN architecture is well suited to Sentiment analysis, it can work by learning to identify patterns in text, such as the presence of certain words or phrases that are associated with positive or negative sentiment.

The following architecture representation for each model is made based on models which were implemented as a code. The structure is following the code representation which can be performed as general steps which are needed to be done in case to succeed in training models.

The following table of architecture representation is demonstrating which steps do Sentiment Analysis model following to achieve a good confidence prediction rate performed on input:

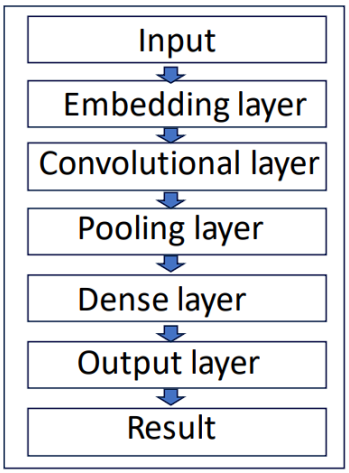


Figure 3.1. Sentiment analysis architecture

And NLP text summarization architectures, particularly those based on Transformer models, have demonstrated superior performance and have become the state-of-the-art approach. It is well suited for abstractive and extractive summarization.

The following table of architecture representation is demonstrating which steps does Extractive Summarization model based on transformers and pipelines follow:

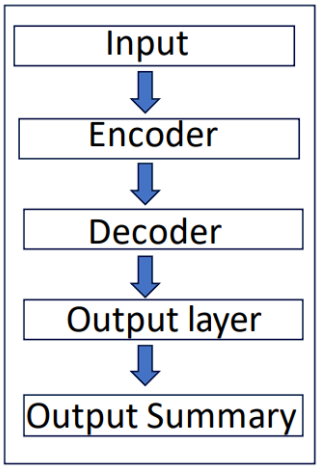


Figure 3.2. Extractive architecture

The following table of architecture representation is demonstrating which steps does Abstractive Summarization model follows:

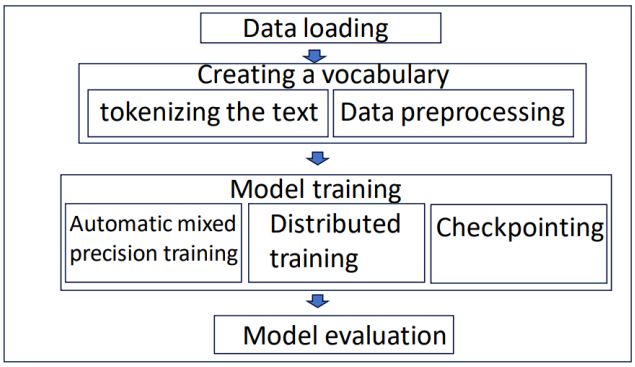


Figure 3.3. Abstractive architecture

As we can see all models have something in common. The main and most important step for each model is to preprocess the input data from a human-readable into a machine-readable one to help the model understand what text is represented based on which perform training. In the next step we are gonna mainly focus on it.

3.2. Preprocessing

Data preprocessing is a crucial step in preparing a dataset for model usage. It involves transforming raw data into a format that is suitable for machine learning models. The preprocessing steps typically include cleaning the data by removing irrelevant characters and special symbols, as well as performing other necessary transformations. Let's explore the steps involved in data preprocessing for machine learning models at the project. Even more these steps are common for CNN and NLP architectures, it's important to transform human readable data into machine-readable format.

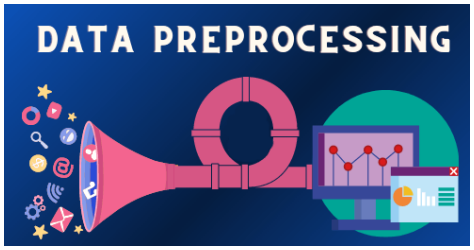


Figure 3.4. Preprocessing

3.2.1. Data Cleaning:

- Removing irrelevant characters: This step involves removing unnecessary characters from the data, such as punctuation marks, and special symbols that do not contribute to the model's understanding.
- Handling missing values:: If the dataset contains missing values, they need to be addressed. This can be done by either removing the rows or columns with missing values or filling in the missing values with appropriate techniques like mean imputation or regression imputation.

3.2.2. Train-Test Split:

- Splitting the dataset into training and testing sets allows for the model's evaluation. Typically, a certain percentage of the data is used for training, while the remaining portion is used for testing the model's performance.

3.3. Abstractive model

The Abstractive Summarization model, used to generate a summary of input text, it utilizes a combination of algorithms and techniques. Each of the methods or components serves a specific purpose in the overall system and contributes to the functionality and performance of the text summarization task. Some of the key algorithms and techniques employed in the Abstractive model are:

- T5 Model: The T5 model is the main algorithm used in this code. It is a transformer-based model pre-trained on a large corpus of text data, fine-tuned for text summarization.
- AdamW: An optimization algorithm based on adaptive gradient descent, commonly used for fine-tuning transformer models.
- LightningModule: The SummaryModel class is a subclass of the PyTorch Lightning Module. It defines the architecture and forwards pass of the T5 model, as well as the training, validation, and testing steps. It also configures the optimizer and loss function for training.

- Trainer: The Trainer class from PyTorch Lightning is used to configure and orchestrate the training process. It takes care of running the training loop, validating the model, and handling various training settings, such as the number of epochs, batch size, and GPU acceleration.

PyTorch Lightning simplifies the training and evaluation process of deep learning models.PyTorch provides seamless GPU acceleration, allowing deep learning models to take advantage of powerful GPU resources. This enables faster training and inference times, making it ideal for working with large datasets and complex models.

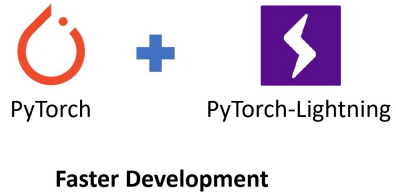


Figure 3.5. Pytorch

The workflow diagram of the Abstractive Summarization T5 model directly shows which exact fine-tuning steps the model followed for the training process. It can be seen that an un-ready dataset must pass through some preparation steps before it can start to interact with models' training steps and be built.

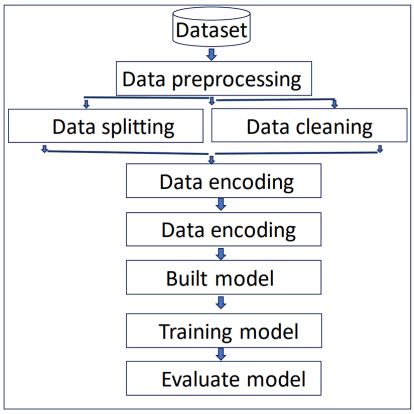


Figure 3.6. Abstractive Summarization Workflow

3.4. Extractive model

The pipeline and Transformers library provides a high-level interface for using pre-trained models in natural language processing tasks. BART is one of the models available in the Transformers library. Hugging Face provides an open-source platform that hosts a wide range of pre-trained models, including BART and more. These models can be easily accessed and used through the Transformers library. Hugging Face's models have achieved state-of-the-art performance on various NLP tasks and are widely adopted by researchers and developers.

```
Require transformers library
Require pipeline from transformers
summarization_pipeline ⬅pipeline("summarization", model="bart")

Require input_text, maxlensum, minlensum

summarybert ⬅ summarization_pipeline(input_text, max_length=maxlensum, min_length=minlensum,
do_sample=False)
summout ⬅ summarybert[0]['summary_text']

Return summout
```

Figure 3.7. BART model pseudocode

The Transformers library, developed by Hugging Face, offers a powerful and user-friendly interface for working with pre-trained models. It provides a unified API, allowing users to perform tasks such as text classification, named entity recognition, question answering, and text summarization. The library includes functionality for tokenization, model loading, fine-tuning, and inference.

The BART model, used for text summarization, utilizes a combination of algorithms and techniques. Some of the key algorithms and techniques employed in the BART model are:

- Transformer Architecture: BART is based on the Transformer architecture, which consists of self-attention mechanisms and feed-forward neural networks. Transformers enable capturing long-range dependencies and contextual relationships in the input text, making them effective for tasks like text summarization.
- Tokenization: BART relies on tokenization to convert text inputs into a sequence of tokens that the model can process. It uses specialized tokenizers, such as the BART tokenizer, to split the input text into meaningful units like words or subwords. This tokenization process enables efficient encoding and decoding of the text during training and inference.

- Encoding: The preprocessed text is encoded using the BART model. BART utilizes a transformer-based encoder-decoder architecture, where the input text is encoded into a numerical representation that captures the semantic meaning of the text.
- Decoding: The encoded representation is then decoded to generate a summary. The decoder part of the BART model takes the encoded representation and generates a condensed summary by predicting the most relevant and salient information from the original text.

The workflow diagram shows generally steps how exactly the model is built. It can be seen that an un-ready dataset must pass through some preparation steps before it can start to interact with models' training steps.

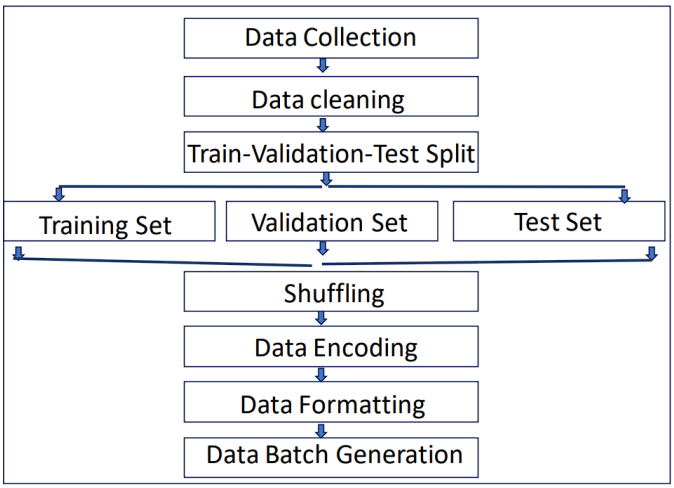


Figure 3.8. General Extractive Summarization Workflow

3.5. Sentiment Analysis

The Sentiment Analysis model, used for text classification, utilizes a combination of algorithms and techniques. Some of the key algorithms and techniques employed in the Sentiment model are:

- Sentiment Analysis is a multi-class classification model, it is trained by applying to the output layer to produce a probability distribution over the classes for multi-class classification.
- TensorFlow provides a robust and flexible platform for implementing sentiment analysis models based on CNN architectures. Its rich set of features, GPU support, and

community support make it a popular choice among researchers and practitioners in the field of natural language processing.

- Sentiment Analysis model is built on the Keras framework.
  - Keras is an open-source deep learning framework written in Python. It provides a high-level and user-friendly interface for building and training neural networks.

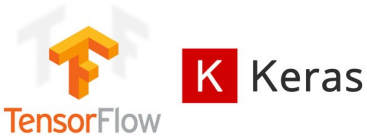


Figure 3.9. Keras and TenserFlow

- Adam Optimizer: It is an adaptive optimization algorithm that is widely used for training neural networks. It adjusts the learning rate during training based on past gradients, making it suitable for a wide range of applications.

And lastly this is workflow diagram for Sentiment analysis:

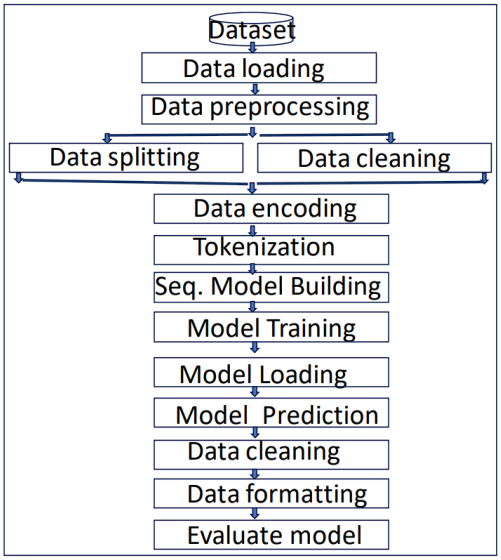


Figure 3.10. Sentiment Analysis Workflow

3.6. Integration and Deployment

After all, goals are achieved for each model, tokenizers, weights, and configuration was saved into pickle format and loaded to google drive to be easily accessible thought the web.Then pickle files were downloaded to the local machine and connected to the GUI application part

Once the pickle files are downloaded to the local machine, they can be loaded into the GUI application using appropriate libraries or frameworks. For example, if the application is built using Python and a library like Tkinter, the pickle files can be imported and deserialized within the application code. This allows the application to access the trained models and perform tasks based on user inputs or data processing requirements.

To ensure easy access to the models through the web, the pickle files were uploaded to Google Drive. By storing the files in the cloud, they can be easily accessed. The pickle format is used to save the eded.

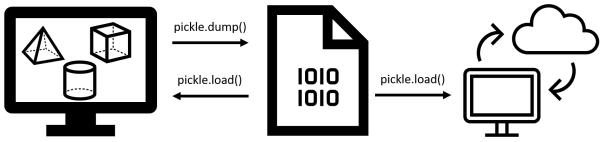


Figure 3.11. Pickle save and load

To utilize the models in a GUI application, the pickle files were downloaded from the cloud and connected to the application's GUI components. This integration allows the application to utilize the pre-trained models for tasks such as text classification, sentiment analysis, or text summarization.

By using pickle files, the models can be efficiently stored, shared, and loaded into applications without the need for retraining or rebuilding the models from scratch. This approach simplifies the deployment process and enables easy access to the trained models for inference or prediction tasks within the GUI application.Pickle files can enable the application to leverage the power of pre-trained deep learning models, enhancing its functionality and providing valuable insights or predictions to the end-users.

4. DESIGN

The user interface of the text summarization and sentiment analysis application was developed using Python, a powerful and efficient programming language that provides an easy and fast solution for building graphical interfaces. Python's extensive library support, including frameworks like Tkinter or PyQt, enables the creation of intuitive and user-friendly interfaces.

The application incorporates four different models, each serving a specific task related to text summarization and classification. The models were trained using Python and then loaded to the cloud, ensuring seamless accessibility and utilization within the application or future basis for other applications.

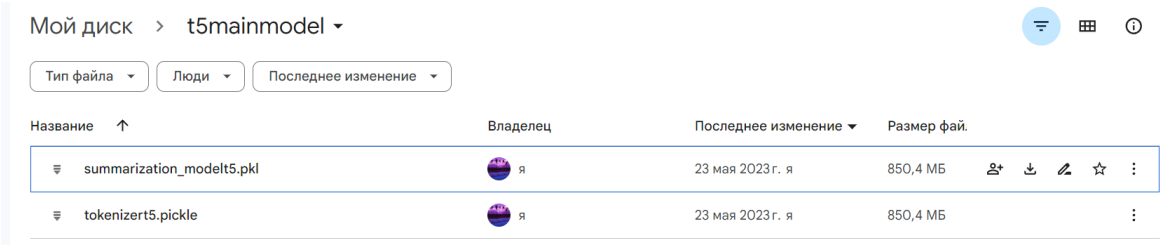


Figure 4.1. Saved model

Before the user can obtain generated summary, the generated text by an Abstractive or Extractive approach needs to be passed through a sentiment analysis model. One important addition is the incorporation of a sentiment analysis model. Before the generated summary is presented to the user, the application passes the generated text, whether obtained through abstractive or extractive approaches, through the sentiment analysis model. This analysis aims to evaluate the polarity and relevance of the text. By considering sentiment analysis, the application can provide a more comprehensive understanding of the summarized content. The flexibility of model selection empowers users to tailor the summarization and sentiment analysis process according to their needs. They can experiment with different combinations of models and evaluate the results to find the most suitable approach for their use case.

When a user interacts with the application, the process begins by providing a text input through the interface. The application offers a default parameter. Generated text summary will be set to the minimum and an abstractive summarization model will be used as a default parameter for the beginning of the operation.

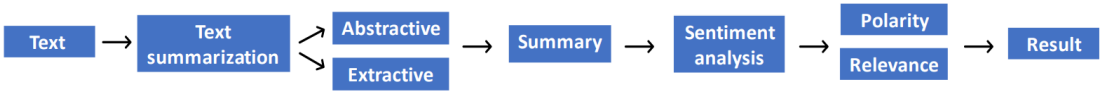


Figure 4.2. Model's interaction

To further decorate consumer enjoyment, the application also consists of functions that improve the usability and comfort of the text summarization and class process.

One such characteristic is the ability to customize the parameters of the summarization process. While default values are supplied to simplify the consumer's interaction, superior customers have the choice to regulate these parameters in keeping with their specific needs. This customization can include adjusting the length of the generated summary or selecting a different summarization model, such as the extractive approach.

The application maintains a cohesive connection between the various functionalities, even though it might not be immediately noticeable during the initial user experience. Once the input text is provided, the application processes the request and generates the desired output, which is displayed in the Result box. To make the application more accessible, it supports multiple input formats.

Users can:

- Send input text directly into the interface, but they also have the option to upload text files from local machine. This versatility accommodates different user preferences and workflows.

Users have multiple options for interacting with the results:

- It is possible to choose to refresh the application and start a new execution by clicking the Clear button and providing an input text field with any textual data. This action resets the output and input boxes, preparing them for fresh input.

- It is also possible easily copy the results to the clipboard or download them to their local machine to any direction using the Copy and Download buttons, respectively.

The application's interface offers a seamless and intuitive experience for users, enabling them to interact with the text summarization and sentiment analysis classification models effortlessly. By incorporating features such as customizable parameters, convenient result handling options, and a user-friendly interface, the application aims to provide an efficient and satisfactory user experience, the text summarization and sentiment analysis classification application becomes a versatile tool that caters to a wide range of user requirements. It aims to streamline the process of obtaining accurate summaries and sentiment analysis results while providing users with flexibility and control over the parameters and inputs.

The complete application which implements the connection between a GUI part with absorbed two summarization and two sentiment analysis deep learning models can be seen from the following UML Use Case diagram:

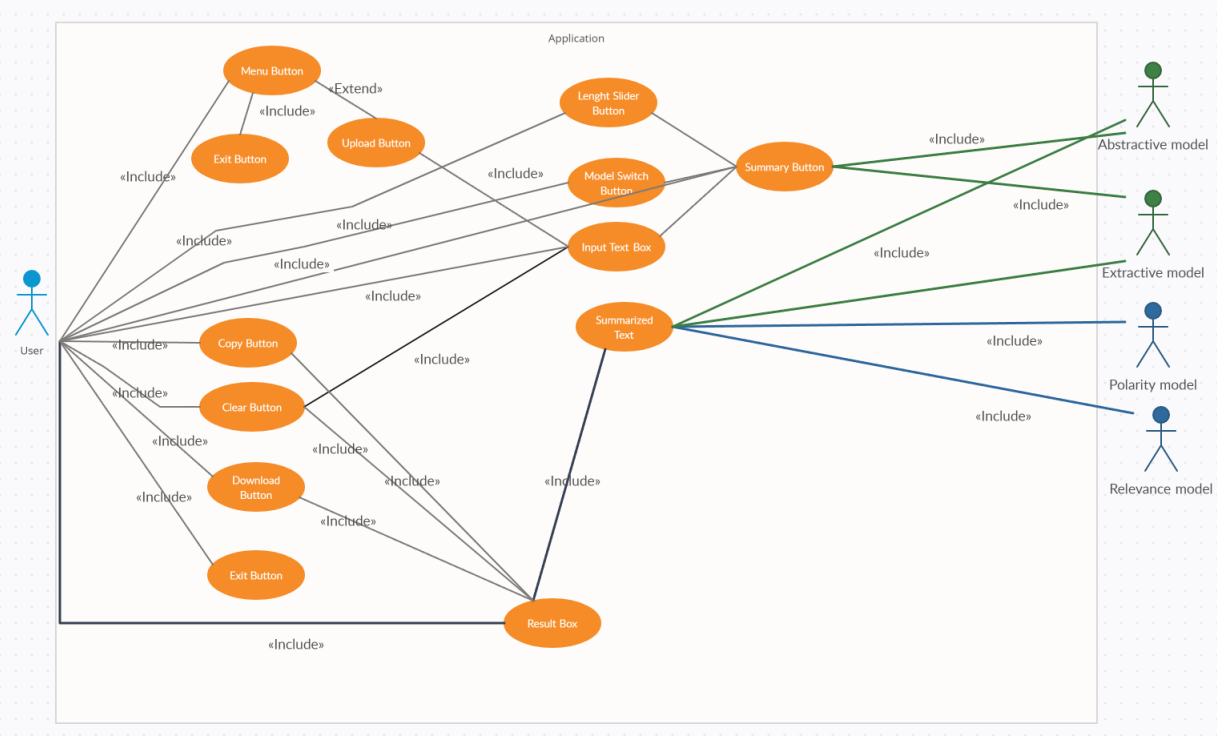


Figure 4.3. Application UML Use Case Diagram

5. IMPLEMENTATION

Firstly models are downloaded from Google Drive and connected with a GUI implementation of the text summarization and classification. In the application's main interface screen users can communicate with several buttons that allow them to generate and classification desire text. One of the most important buttons is the ”Summarize” button, which is responsible for executing the model and generating the desired result. Whenever the user interacts with this specific button it is triggering models, if no change was performed besides the Summarization button, the default setting for the model will be chosen by the application.

5.1. User Interface

The input interface of the application allows users to enter text directly into a text box or upload a text file by using the Windows search system field which appears right after the ”Menu” button was pressed. Users can type or paste their desired text into the text box or use the file upload feature to select a specific text file from their computer.

In case a user wants to interact with additional buttons on the screen right after entering the text, these buttons may include options for customizing the summarization process, such as setting the length of the summary or specifying any additional parameters.

After the user's decision on customization is performed, the user can click the ”Summarize” button to receive a generated text from an application that passes it through different mounted deep learning models. The text is generated based on its learning patterns and knowledge.The generated text then displayed on the user's application's interface

This user interface represents a tool for abstractive text summarization with text classification, where both tasks are performed simultaneously to generate the desired result. The interface includes various elements that allow users to customize the summarization process.

One of the key features is the ability to specify the length of the summary. Users can choose a desired summary length, determining the level of detail they want in the generated summary. Additionally, there is an option to set the length of penalty, which can control the trade-off between brevity and informativeness in the summary.

The main input area of the tool allows users to enter the text they want to summarize.

They can type or paste the text directly into the input box or upload a text file using the provided file upload button. This flexibility enables users to conveniently provide the input text from different sources.

Once the user has entered the input text, they can initiate the summarization process by clicking the ”Summarize” button. The tool then passes the necessary arguments, including the chosen summary length, length of penalty, and the input text, to the underlying deep learning model.

The model processes the input text using abstractive text summarization techniques, which involve understanding the context, generating new sentences, and capturing the key information. Simultaneously, the model also performs text classification to extract relevant information and classify the input text into specific categories or topics.

After the model completes its processing, it sends back the generated summary text, which appears in the ”Result” box on the interface. Users can read the summary and assess its quality and relevance based on their requirements.

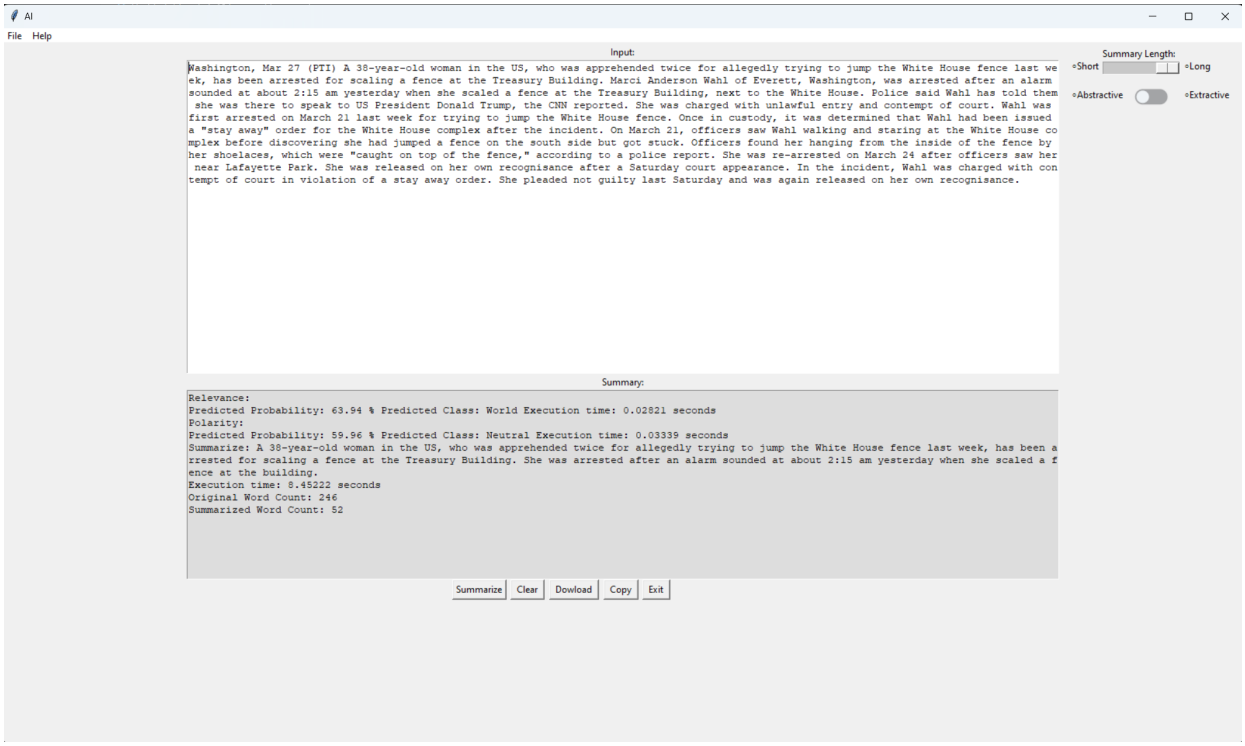


Figure 5.1. User Interface 1

This user interface also supports extractive summarization, where the generated summary focuses on extracting the most important patterns and sentences from the input text to provide the user with the main idea. Similar to Abstractive summarization, this model approach allows users to customize the summarization process by choosing the maximum and minimum length for the generated summary by the slider button.

Users can enter the text they want to summarize in the input box. Once the input text is provided, users can click the ”Summarize” button to trigger the extractive summarization and classification models. The interface passes the necessary arguments, such as maximum and minimum lengths, along with the input text, to the underlying deep learning model.The model utilizes extractive summarization techniques to analyze the input text and identify the most important patterns and sentences. It selects the sentences that best capture the main idea and significance of the text while considering the specified length constraints.

After the model completes its processing, the generated summary text, which consists of the extracted sentences, appears in the ”Result” box on the interface. Users can read the summary to gain an understanding of the main points and key information present in the input text.

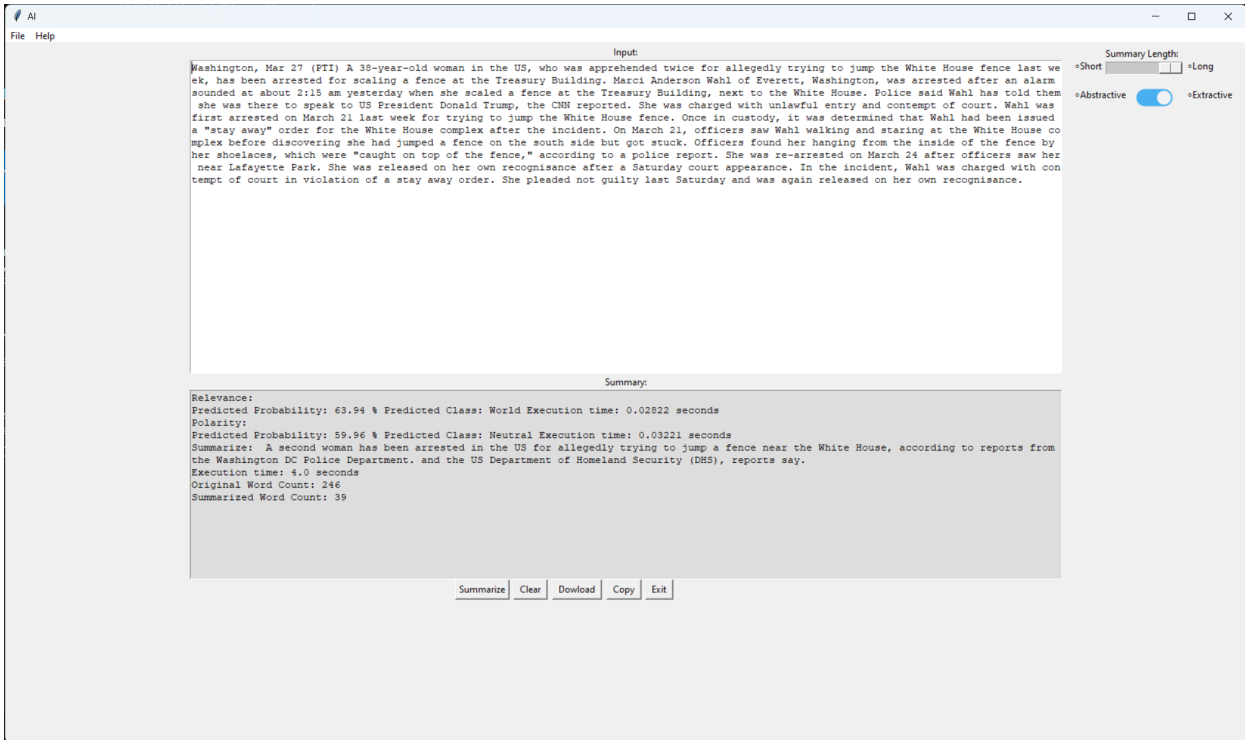


Figure 5.2. User Interface 2

5.2. Downloading Results

After generating the output using the Summarization and Sentiment Analysis models in the previous step, the next logical extension is to provide the user with the option to save the summary if they find it satisfactory. This step can be considered as optional.

To facilitate this functionality, the GUI application can provide a ”Download” button that allows the user to save the generated summary as a text file. The application can prompt the user to choose the desired location and specify the name of the file, including the appropriate file extension (such as .txt).

By enabling the option to save the summary locally, users can store important or valuable summaries for later use, share them with others, or integrate them into their own documents or reports.If the generated summary is considered weak or unsatisfactory, the application can provide an alternative by allowing the user to generate a new summary. This flexibility ensures that the user has the opportunity to refine or improve the summary based on their preferences or specific requirements.

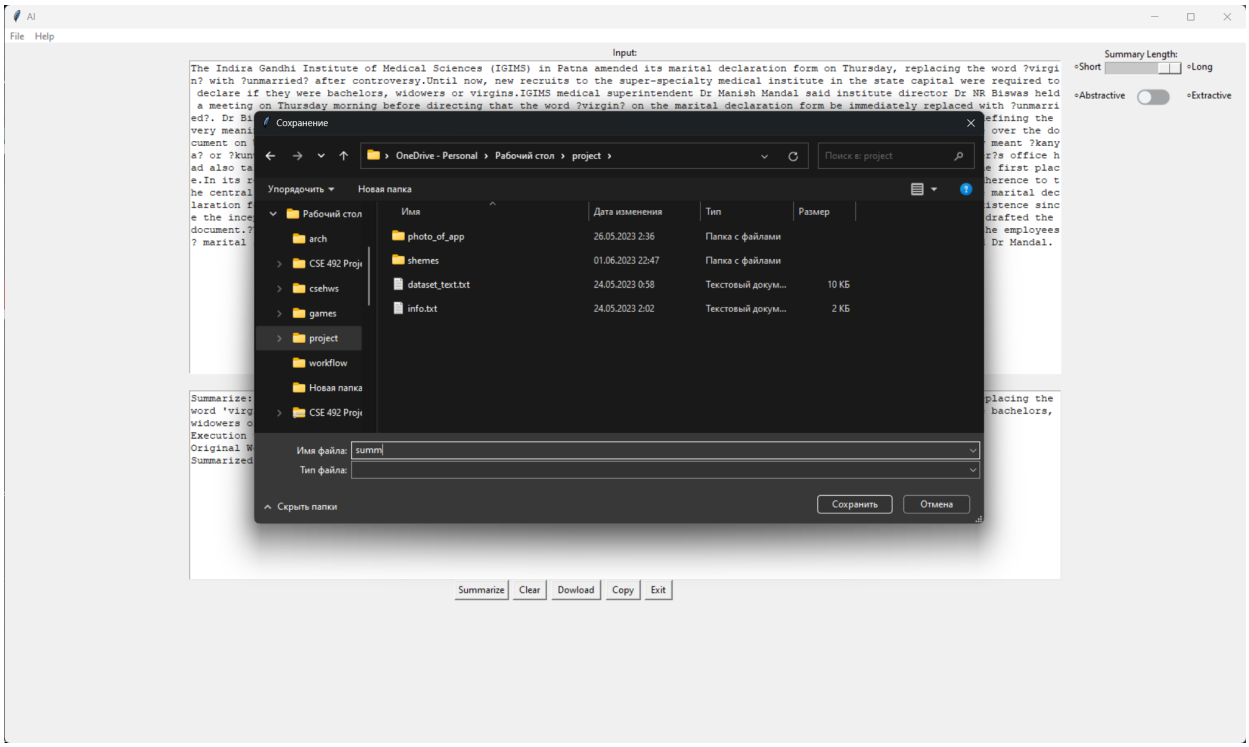


Figure 5.3. Downloaded Result file

5.3. Uploading file

In addition to generating summaries and performing sentiment analysis on provided text, the GUI application can further enhance its functionality by allowing users to load files located on their local machines. This can be achieved through the integration of the Windows search tool, accessed by pressing the Menu button. This step also can be considered optional to perform when the summarization and sentiment analysis interaction with aplication happens.

By utilizing the Windows search tool, users can easily navigate their local file system and select the desired file to load into the application. The GUI application can provide a ”Open” button.

By allowing users to load files from their local machines, the GUI application enables them to work with a wide range of textual data, such as articles, or reports.This flexibility expands the scope of the application’s functionality beyond just generating summaries and performing sentiment analysis on user-provided text.

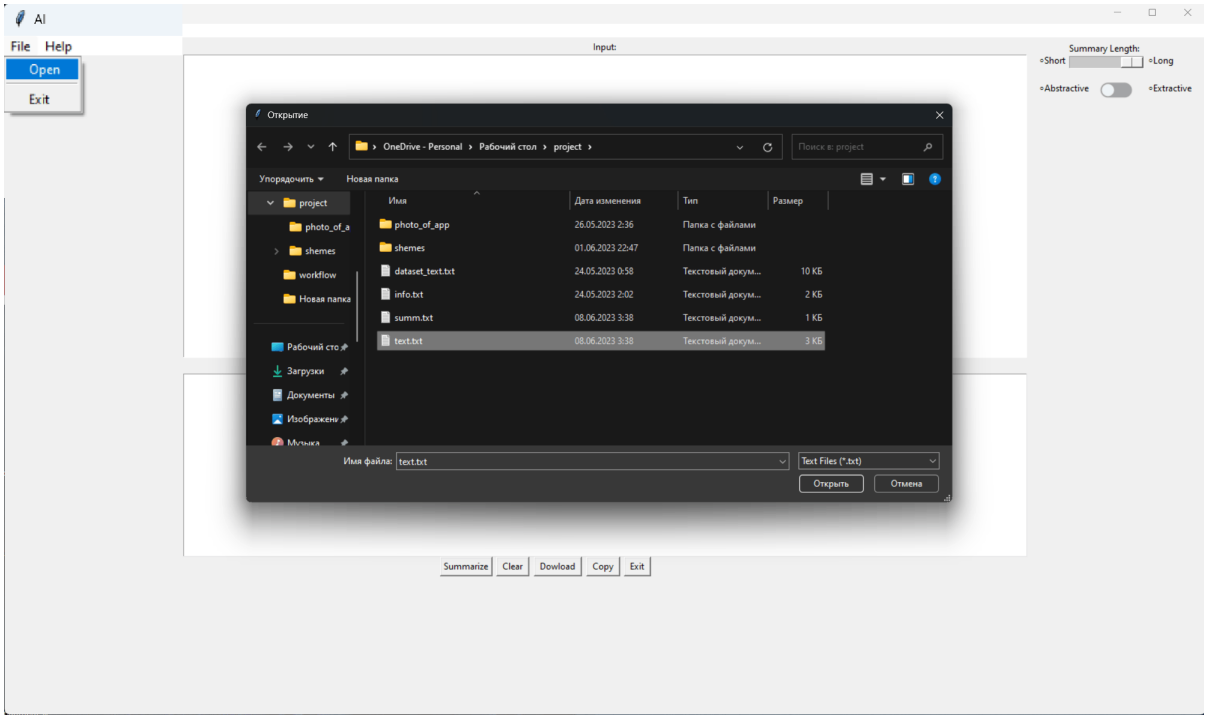


Figure 5.4. Upload Menu Button

6. TEST AND RESULTS

6.1. Evaluation Setup

- *Preprocessing:* Firstly input text data is preprocessed by tokenizing, cleaning, and normalizing to ensure consistency and compatibility with models.
- *Model Configuration:* For text summarization pre-trained model architecture is used, Natural Language Processing (NLP) is well suited to abstractive text summarization and it has been chosen to generate a summary.T5-based model have been fine-tuned and trained with a news dataset. On the other hand, the transformer-based BART model is used for extractive text summarization, it was trained on a large dataset of text and code. Both model have a perfect performance in the field of text summarization. For sentiment analysis, a convolutional neural network (CNN) is used to extract features from the text. This model can identify patterns and phrases in the text and be able to classifier to predict the sentiment of a text.

6.2. Datasets

- *Model Configuration:* To train models five datasets were used.

  Text Summarization:

- *Dataset for Abstractive Summarization:* The News The summary dataset on Kaggle is used for abstractive text summarization and it is a collection of news articles that have been summarized by a human it contains 4515 examples, and has been classified into five categories: Author, Headline, URL, Short text, and Complete article. Only Short text and Complete articles are used to train the model.
- *Dataset for Extractive Summarization:* The XSum dataset from HuggingFace is a dataset of news articles that contains 226,711 examples, each example has been classified into two categories: Document and Summary. The CNN DailyMail dataset is the second dataset that was used to train the model. It consists of 300,000 examples which are classified into two categories: Document and Summary.

  Sentiment Analysis:

- *Dataset for Polarity:*The Twitter Entity Sentiment Analysis from Kaggle is the dataset of tweet collections that have been labeled with their sentiment that have been classified

into four categories: Positive, Negative, Neutral, and Irrelevant. The dataset contains 74,000 examples.

- *Dataset for Relevance:* The AG News dataset from HuggingFace is a collection of news articles that have been classified into four categories: World, Sports, Business, and Sci/Tech. The dataset contains 120,000 training examples and 7,600 test examples. Each example consists of a title and a description(label). The titles and descriptions are tokenized and padded to a maximum length of 128 tokens.

6.3. Evaluation Metrics and Results

- For the evaluation of text summarization BERTScore is used, it evaluates the similarity between the generated summary and the reference summary using contextualized word embeddings. The higher score of similarity based on BERTScore indicates better similarity. BERTScore consist of Precision,Recall and F1 values.
- *The precision* is correctly predicted ratio of positive classes to all predicted positive classesand. It can be expressed as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- *The Recal* is correctly predicted positive classes ratio to all actually existing positive classes.It can be expressed as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- *F1-score* is basically a combination of precision and recall ratio's into a single metric and it is expressed as follow:

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FN} + \text{FP})}$$

The following table is intended to help to understand how each of the model is performing.

Table 6.1. Model performance and accuracy

Score	Interpretation
0-20	Bad
20-40	Fair
40-60	Good
60-80	Excellent
80-100	Perfect

From this table it can be seen that the Abstractive text summarization similarity score is exorbitantly high and it is indicated that the finite-tuned model can generate a summary that is very close to the dataset it was trained on.

Table 6.2. Abstractive model accuracy

Sumary	Precision	Recall	F1
1	91	90	90
2	93	92	90
3	92	92	92
4	86	81	83
5	93	92	93
AVG:	91	89.4	90

Results of *Abstractive* and *Extractive* Text Summarization can be considered perfect or excellent if the result of BERTScore is within the confines of between 40 and 100. Even if the result score is around 30, it is considered to be a good score for extractive and abstractive summarization. This means that the generated summary is similar by 30 percent to a human-generated summary.

It is also crucial to take into account the original and summarized word count, if generated summary word count is far less than the original input we can consider that model performs well. For Text summarization and sentiment analysis, time will be measured to calculate the performance of each model by taking the average results of each class of polarity and relevance model.

On the other hand Extractive text summarization gives a clear instance that it can extract all important phrases and sentences to create a good result summary.

Table 6.3. Extractive model accuracy

Sumary	Precision	Recall	F1
1	91	89	90
2	81	80	81
3	83	81	82
4	80	76	78
5	77	83	84
AVG:	82.4	81.8	83

For Abstractive text summarization model we can obtain following table:

Table 6.4. Abstractive model performance

Test	MinTime	MinLen	OriginalWorldCount	MaxTime	MaxLen
1	7.4	39	364	8.8	57
2	7.3	25	335	9.7	43
3	7.0	40	526	7.8	58
4	8.4	46	404	9.7	62
5	8.1	35	453	7.4	55
AVG:	7.6	37	394.4	8.9	55

For Extractive text summarization model we can obtain following table:

Table 6.5. Extractive model performance

Test	MinTime	MinLen	OriginalWorldCount	MaxTime	MaxLen
1	4.6	28	450	10.8	88
2	6.6	35	698	12.1	104
3	6.0	33	738	12.5	100
4	5.2	34	414	10.7	97
5	5.9	30	705	12.3	94
AVG:	5.6	32	601	11.68	96.6

However, it is always important to note that BERTScore scores are only a measure of similarity between two sentences, and it does not necessarily indicate the quality of the summary. A summary with a high BERTScore score may not be accurate or informative. In this case, to consider the accuracy, informativeness, and fluency of the summary human evaluation is the one of best possible solutions to evaluate summary. In a human evaluation, a group of human judges is asked to rate the quality of a set of generated summaries at a rate from one to five.

Table 6.6. Human evaluation

Test	1(bad)	2(fair)	3(good)	4(excellent)	5(perfect)
MYKOLA ABLAPOKHIN					v
Meylis Hangeldiyev					v
Ekingül Bahadır				v	
Aisel Vlad					v
Berat Berk Erkan					v
Yiğit Şengüloğlu					v
Aydın Deniz Kadıoğlu				v	
Bilal Cilga					v
Gokhan Keray					v
Gorkem Ozgul				v	
Kaan Ozcan					v
GBerkay Eraydin				v	
Melisa Timiski					v
Celil Arslan			v		
Dorukan Zaren					v
Bahadir Kurda				v	
Ismail Cem Unver					v
Ersa Bingol				v	
Umut Ege Ozkan					v
Mehmet Akif Petek				v	
Bekir Ozcelik					v

Average for human evalution score is 4.57 out of 5

Sentiment analysis consists of two well-trained classification models on different public datasets. Each model was tested on twenty examples from a origin dataset which makes five examples for each model class. Their confidence rate is between 80 and 100 which is considered as excellent. Also, the average time of prediction is extremly low and it is fantastic.

We can obtain an understanding how good is Polarity sentiment analysis model from the corresponding table with four classes which are tasted on Time performance and average performance of the model:

Table 6.7. Polarity Results

CLASS	TIME	AVG%
IRRELEVANT	0.0325	81.0%
NEGATIVE	0.0324	95.4%
POSITIVE	0.0352	69.74%
NEUTRAL	0.0328	93.6%
AVG:	0.0337	84.97%

As same as for Polarity model we can obtain an understanding how good is Relevance sentiment analysis model from the corresponding table with four classes which are tasted on Time performance and average performance of the model: For Relevance classification model we obtain following table:

Table 6.8. Relevance Results

CLASS	TIME	AVG%
WORLD	0.0317	99.9%
SPORT	0.0324	100%
BUSINESS	0.0394	99.8%
TECH	0.3234	99.8%
AVG:	0.0339	99.91%

7. CONCLUSION

It can be seen that Text summarization and Sentiment analysis have good performance rates and can generate summaries and certificate classes fast and even the rate of accuracy and prediction percentage are extremely high and such a good basis and starting point can be reasonable to use this approach for other applications and projects which are based on deep learning.From overall work done it is also can be seen that wide range of techologies were used to implement all steps in case to obtain good generated summary and confidence of sentiment analysis model.

7.1. Future Work

To improve the fine-tuned t5-base model by training it with different language datasets in case to have a powerful summarization tool for users to be able to choose any language and length they want to gain their requirement goal. Since this model exists not only in the base version, it would be wise to train this with the model with a larger amount of patterns which will allow capturing more complex patterns and dependencies in the input text which will lead to increased performance on various natural language processing tasks. The only challenge that can be faced is that increasing the amount of capturing patterns will lead to a need for great computational resources and time for training. Same for sentiment analysis, I would like to have more extravagant datasets such as to be able to deal with any given language. Also by improving my models with new datasets, it would be great to increase the performance by reducing the computation time.

Bibliography

[1] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[2] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67,

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[4] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *CoRR*, vol. 1910.13461, pp. 7871–7880, 2019.

[5] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014, pp. 655–665.

[6] Quillbot. "Quillbot - ai paraphrasing tool." (2023), [Online]. Available: `bit.ly/3Xcamit`.

[7] MonkeyLearn. "Monkeylearn - sentiment analysis online." (2023), [Online]. Available: `https://monkeylearn.com/sentiment-analysis-online/`.