DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

MASTER OF SCIENCE

ON

WEB INTELLIGENCE

Extractive Text Summarization with Machine Learning and Natural Language Processing

MASTER'S THESIS

by

DOULAKIS STYLIANOS

Supervisor: Konstantinos Diamantaras
Professor

Thessaloniki, Sep 2021

This page is intentionally left blank.

INTERNATIONAL HELLENIC UNIVERSITY

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

MASTER'S PROGRAM ON WEB INTELLIGENCE

Extractive Text Summarization by using Machine Learning and Natural Language Processing techniques

MASTER'S THESIS

by

DOULAKIS STYLIANOS

Supervisor : Konstantinos Diamantaras
Professor

Approved by the evaluation committee on Choose a date.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Name Surname	Name Surname	Name Surname
Choose an item. I.H.U.	Choose an item. I.H.U.	Choose an item. I.H.U.

Thessaloniki, Choose a date

(Signature)

.....

Stylianos Doulakis

Click here to enter text.

Abstract

The goal of this Thesis is to give the audience of what text summarization is and the theoretical background based on bibliography of different models and techniques could be used for text summarization challenges. Apart from the theoretical reference of the models we will leverage the Bidirectional Encoder Representation from Transformers (BERT) which is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google [1] to create a web application that will help simple users with no experience to the field to summarize their texts.

Keywords: NLP, BERT, ML, Text Summarization

This page is left intentionally blank.

Table of Contents

1 Introduction 1

1.1 Text summarization 1

1.2 Extractive vs Abstractive text summarization 1

1.3 Extractive text summarization with BERT 2

1.3.1 Contribution 2

1.4 Chapter organization 2

2 Related work 4

2.1 Natural Language Processing 4

2.1.1 What is natural language processing? 4

2.1.2 How does natural language processing *work? 4*

2.1.3 Techniques and methods of natural language processing 5

2.1.4 What is natural language processing used for? 7

2.1.5 Benefits of natural language processing 9

2.1.6 Challenges of natural language processing 9

2.1.7 The evolution of natural language processing 10

2.2 Machine Learning 11

2.2.1 Introduction 11

2.2.2 What is machine learning? 12

3 Theoretical background 16

3.1 Bidirectional Encoder Representation from Transformers (BERT) 16

3.1.1 Background 16

3.1.2 How BERT works 17

3.1.3 Masked LM (MLM) 17

3.1.4 Word Masking 18

3.1.5 Next Sentence Prediction (NSP) 18

3.1.6 How to use BERT (Fine-tuning) 19

3.1.7 Different Variations of BERT 20

3.1.8 Takeaways 21

3.1.9 Conclusion *22*

4 High level overview of the Web Application Logic **23**

4.1 Web Application 23

4.1.1 Front-end *23*

4.1.2 Back-end *24*

4.2 Summarization Process 24

5 The Solution **25**

5.1 The Application 25

5.1.1 Project Structure *25*

5.1.2 Text Summarizer *31*

6 Evaluation **33**

6.1 Choose a Model 33

6.2 GLUE as Evaluation system 34

6.2.1 Single Sentence Tasks *34*

6.2.2 Inference Tasks *34*

6.3 Manual Tests 34

6.3.1 Example 1 ..... *35*

6.3.2 Example 2 ..... *38*

6.3.3 Example 3 ..... *39*

7 Technical details **41**

7.1 Libraries 41

7.1.1 Python and Flask *41*

7.1.2 Bootstrap *42*

7.1.3 NLTK *43*

7.2 Platforms and Coding Tools 43

7.2.1 PycharmCE *43*

7.2.2 Python pip *44*

7.2.3 Python venv *44*

8 Epilogue **45**

8.1 Summary and conclusions 45

8.2 Future extensions 45

8.2.1 Application performance cache *45*

8.2.2 Web Application Programming Interface (API) *46*

8.2.3 Integration with a Queue *46*

8.2.4 Load Balancing *47*

9 Bibliography **49**

1

Introduction

1.1 Text summarization

With the exponential growth of the Internet content and the number of web pages there is a real need for automatic text summarization (we will simply call it as text summarization). The definition of a text summary could describe as “As the text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that” [2].

As text summarization we can define the task that produces a fluent summary of the original text by keeping the main context and meaning in place [3].

1.2 Extractive vs Abstractive text summarization

In order to automate the task of text summarization there are two important techniques we could follow. The first one which is extractive text summarization method relies on a two-phase process where in the first step we can extract textual elements from the original text like keywords, clauses, sentences or even paragraphs by using techniques of linguistic or other statistical models. The next step is to use the extracted elements put them in the right order to provide the final summary of the text.

The second approach is the abstractive text summarization method where we try to understand the original text provided and then re-phrase it in a way that is understandable and create a summary. The main differentiation is that the abstractive may or may not include sentences or words from the original text compared to extractive where the final text has parts of the most important elements of the original text [4].

1.3 Extractive text summarization with BERT

Specifically for this Thesis we created a web application with a simplified User Interface (UI) to enable users to summarize their texts written in English language. For the model behind the web application, we leveraged the Bidirectional Encoder Representation from Transformers (BERT) which is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google [1]. The overall challenge is to integrate the model with the web UI and configure it in a way to reply in a meaningful time the results to the user who requests a text to be summarized.

1.3.1 Contribution

The thesis contribution is summarized as follows:

1. We studied and researched the field of extractive text summarization using the BERT Model.
2. We implemented a Web application with a user-friendly simple UI to interact with BERT.
3. We reviewed the performance of different variations of BERT for text summarization that fits out problem.
4. We configured the model in a way to serve the needs of a web application.
5. We have integrated some pre-trained BERT algorithms into a prototype web application. To help users summarize their texts.

1.4 Chapter organization

In this Chapter we start with a brief overview of the broader topic of text summarization and technologies behind, in Chapter 2 we will refer to other related work in the field of extractive text summarization done by others, Chapter 3 is all about the theoretical background of the different models we could potentially leverage to resolve the problem, Chapter 4 is about the analysis of the model we used, Chapter 5 is how we solved the problem, Chapter 6 is about the

evaluation of our solution, Chapter 7 is related to the technical details, technologies and tools we used during the execution, Chapter 8 is all about learning and what we can improve for the future and in Chapter 9 you can find all the references to bibliography, papers and other resources.

2

Related work

In this chapter we briefly write the thematic areas in which we have discovered relevant work and explain why these areas are relevant to the thesis.

2.1 Natural Language Processing

Here we present works describing available technologies/models/methodologies in "Thematic Area 1" which are related to the thesis.

It is important to stress the differences and similarities in relation to your own thesis.

2.1.1 What is natural language processing?

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written -- referred to as natural language. It is a component of artificial intelligence (AI).

NLP has existed for more than 50 years and has roots in the field of linguistics. It has a variety of real-world applications in a number of fields, including medical research, search engines and business intelligence.

2.1.2 How does natural language **processing** *work?*

NLP enables computers to understand natural language as humans do. Whether the language is spoken or written, natural language processing uses artificial intelligence to take real-world input, process it, and make sense of it in a way a computer can understand. Just as humans have different sensors -- such as ears to hear and eyes to see -- computers have programs to read and microphones to collect audio. And just as humans have a brain to process that input, computers have a program to process their respective inputs. At some point in processing, the input is converted to code that the computer can understand.

There are two main phases to natural language processing: data preprocessing and algorithm development.

Data preprocessing involves preparing and "cleaning" text data for machines to be able to analyze it. preprocessing puts data in workable form and highlights features in the text that an algorithm can work with. There are several ways this can be done, including:

- Tokenization. This is when text is broken down into smaller units to work with.
- Stop word removal. This is when common words are removed from text so unique words that offer the most information about the text remain.
- Lemmatization and stemming. This is when words are reduced to their root forms to process.
- Part-of-speech tagging. This is when words are marked based on the part-of speech they are -- such as nouns, verbs and adjectives.

Once the data has been preprocessed, an algorithm is developed to process it. There are many different natural language processing algorithms, but two main types are commonly used:

- Rules-based system. This system uses carefully designed linguistic rules. This approach was used early on in the development of natural language processing, and is still used.
- Machine learning-based system. Machine learning algorithms use statistical methods. They learn to perform tasks based on training data they are fed, and adjust their methods as more data is processed. Using a combination of machine learning, deep learning and neural networks, natural language processing algorithms hone their own rules through repeated processing and learning.

2.1.3 Techniques and methods of natural language processing

Syntax and semantic analysis are two main techniques used with natural language processing.

Syntax is the arrangement of words in a sentence to make grammatical sense. NLP uses syntax to assess meaning from a language based on grammatical rules. Syntax techniques include:

Parsing. This is the grammatical analysis of a sentence. Example: A natural language processing algorithm is fed the sentence, "The dog barked." Parsing involves breaking this sentence into parts of speech -- i.e., dog = noun, barked = verb. This is useful for more complex downstream processing tasks.

- Word segmentation. This is the act of taking a string of text and deriving word forms from it. Example: A person scans a handwritten document into a computer. The algorithm would be able to analyze the page and recognize that the words are divided by white spaces.

- Sentence breaking. This places sentence boundaries in large texts. Example: A natural language processing algorithm is fed the text, "The dog barked. I woke up." The algorithm can recognize the period that splits up the sentences using sentence breaking.
- Morphological segmentation. This divides words into smaller parts called morphemes. Example: The word untestably would be broken into [[un[[test]able]]ly], where the algorithm recognizes "un," "test," "able" and "ly" as morphemes. This is especially useful in machine translation and speech recognition.
- Stemming. This divides words with inflection in them to root forms. Example: In the sentence, "The dog barked," the algorithm would be able to recognize the root of the word "barked" is "bark." This would be useful if a user was analyzing a text for all instances of the word bark, as well as all of its conjugations. The algorithm can see that they are essentially the same word even though the letters are different.
- Semantics involves the use of and meaning behind words. Natural language processing applies algorithms to understand the meaning and structure of sentences. Semantics techniques include:
- Word sense disambiguation. This derives the meaning of a word based on context. Example: Consider the sentence, "The pig is in the pen." The word pen has different meanings. An algorithm using this method can understand that the use of the word pen here refers to a fenced-in area, not a writing implement.
- Named entity recognition. This determines words that can be categorized into groups. Example: An algorithm using this method could analyze a news article and identify all mentions of a certain company or product. Using the semantics of the text, it would be able to differentiate between entities that are visually the same. For instance, in the sentence, "Daniel McDonald's son went to McDonald's and ordered a Happy Meal," the algorithm could recognize the two instances of "McDonald's" as two separate entities -- one a restaurant and one a person.
- Natural language generation. This uses a database to determine semantics behind words and generate new text. Example: An algorithm could automatically write a summary of findings from a business intelligence platform, mapping certain words and phrases to features of the data in the BI platform. Another example would be automatically generating news articles or tweets based on a certain body of text used for training.

Current approaches to natural language processing are based on deep learning, a type of AI that examines and uses patterns in data to improve a program's understanding. Deep learning models require massive amounts of labeled data for the natural language processing algorithm to train on and identify relevant correlations, and assembling this kind of big data set is one of the main hurdles to natural language processing. [17]

Earlier approaches to natural language processing involved a more rules-based approach, where simpler machine learning algorithms were told what words and phrases to look for in text and given specific responses when those phrases appeared. But deep learning is a more flexible, intuitive approach in which algorithms learn to identify speakers' intent from many examples -- almost like how a child would learn human language.

Three tools used commonly for natural language processing include Natural Language Toolkit (NLTK), Gensim and Intel natural language processing Architect. NLTK is an open source Python module with data sets and tutorials. Gensim is a Python library for topic modeling and document indexing. Intel NLP Architect is another Python library for deep learning topologies and techniques.

2.1.4 What is natural language processing used for?

Some of the main functions that natural language processing algorithms perform are:

- Text classification. This involves assigning tags to texts to put them in categories. This can be useful for sentiment analysis, which helps the natural language processing algorithm determine the sentiment, or emotion behind a text. For example, when brand A is mentioned in X number of texts, the algorithm can determine how many of those mentions were positive and how many were negative. It can also be useful for intent detection, which helps predict what the speaker or writer may do based on the text they are producing.
- Text extraction. This involves automatically summarizing text and finding important pieces of data. One example of this is keyword extraction, which pulls the most important words from the text, which can be useful for search engine optimization. Doing this with natural language processing requires some programming -- it is not completely automated. However, there are plenty of simple keyword extraction tools that automate most of the process -- the user just has to set parameters within the program. For example, a tool might pull out the most frequently used words in the text. Another example is named entity recognition, which extracts the names of people, places and other entities from text.
- Machine translation. This is the process by which a computer translates text from one language, such as English, to another language, such as French, without human intervention.
- Natural language generation. This involves using natural language processing algorithms to analyze unstructured data and automatically produce content based on

that data. One example of this is in language models such as GPT3, which are able to analyze an unstructured text and then generate believable articles based on the text.

The functions listed above are used in a variety of real-world applications, including:

- customer feedback analysis -- where AI analyzes social media reviews;
- customer service automation -- where voice assistants on the other end of a customer service phone line are able to use speech recognition to understand what the customer is saying, so that it can direct the call correctly;
- automatic translation -- using tools such as Google Translate, Bing Translator and Translate Me;
- academic research and analysis -- where AI is able to analyze huge amounts of academic material and research papers not just based on the metadata of the text, but the text itself;
- analysis and categorization of medical records -- where AI uses insights to predict, and ideally prevent, disease;
- word processors used for plagiarism and proofreading -- using tools such as Grammarly and Microsoft Word;
- stock forecasting and insights into financial trading -- using AI to analyze market history and 10-K documents, which contain comprehensive summaries about a company's financial performance;
- talent recruitment in human resources; and
- automation of routine litigation tasks -- one example is the artificially intelligent attorney.

Research being done on natural language processing revolves around search, especially Enterprise search. This involves having users query data sets in the form of a question that they might pose to another person. The machine interprets the important elements of the human language sentence, which correspond to specific features in a data set, and returns an answer.

NLP can be used to interpret free, unstructured text and make it analyzable. There is a tremendous amount of information stored in free text files, such as patients' medical records. Before deep learning-based NLP models, this information was inaccessible to computer-assisted analysis and could not be analyzed in any systematic way. With NLP analysts can sift through massive amounts of free text to find relevant information.

Sentiment analysis is another primary use case for NLP. Using sentiment analysis, data scientists can assess comments on social media to see how their business's brand is performing, or review notes from customer service teams to identify areas where people want the business to perform better.

2.1.5 Benefits of natural language processing

The main benefit of NLP is that it improves the way humans and computers communicate with each other. The most direct way to manipulate a computer is through code -- the computer's language. By enabling computers to understand human language, interacting with computers becomes much more intuitive for humans.

Other benefits include:

- improved accuracy and efficiency of documentation;
- ability to automatically make a readable summary of a larger, more complex original text;
- useful for personal assistants such as Alexa, by enabling it to understand spoken word;
- enables an organization to use chatbots for customer support;
- easier to perform sentiment analysis; and
- provides advanced insights from analytics that were previously unreachable due to data volume.

2.1.6 Challenges of natural language processing

There are a number of challenges of natural language processing and most of them boil down to the fact that natural language is ever-evolving and always somewhat ambiguous. They include:

- Precision. Computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured -- or through a limited number of clearly enunciated voice commands. Human speech, however, is not always precise; it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.
- Tone of voice and inflection. Natural language processing has not yet been perfected. For example, semantic analysis can still be a challenge. Other difficulties include the fact that the abstract use of language is typically tricky for programs to understand. For instance, natural language processing does not pick up sarcasm easily. These topics usually require understanding the words being used and their context in a conversation. As another example, a sentence can change meaning depending on which word or syllable the speaker puts stress on. NLP algorithms may miss the subtle, but important, tone changes in a person's voice when performing speech recognition. The tone and inflection of speech may also vary between different accents, which can be challenging for an algorithm to parse.
- Evolving use of language. Natural language processing is also challenged by the fact that language -- and the way people use it -- is continually changing. Although there

are rules to language, none are written in stone, and they are subject to change over time. Hard computational rules that work now may become obsolete as the characteristics of real-world language change over time.

2.1.7 The evolution of natural language processing

NLP draws from a variety of disciplines, including computer science and computational linguistics developments dating back to the mid-20th century. Its evolution included the following major milestones:

- 1950s. Natural language processing has its roots in this decade, when Alan Turing developed the Turing Test to determine whether or not a computer is truly intelligent. The test involves automated interpretation and the generation of natural language as criterion of intelligence.
- 1950s-1990s. NLP was largely rules-based, using handcrafted rules developed by linguists to determine how computers would process language.
- 1990s. The top-down, language-first approach to natural language processing was replaced with a more statistical approach, because advancements in computing made this a more efficient way of developing NLP technology. Computers were becoming faster and could be used to develop rules based on linguistic statistics without a linguist creating all of the rules. Data-driven natural language processing became mainstream during this decade. Natural language processing shifted from a linguist-based approach to an engineer-based approach, drawing on a wider variety of scientific disciplines instead of delving into linguistics.
- 2000-2020s. Natural language processing saw dramatic growth in popularity as a term. With advances in computing power, natural language processing has also gained numerous real-world applications. Today, approaches to NLP involve a combination of classical linguistics and statistical methods.

Natural language processing plays a vital part in technology and the way humans interact with it. It is used in many real-world applications in both the business and consumer spheres, including chatbots, cybersecurity, search engines and big data analytics. Though not without its challenges, NLP is expected to continue to be an important part of both industry and everyday life.

2.2 Machine Learning

2.2.1 Introduction

Machine learning is behind chatbots and predictive text, language translation apps, the shows Netflix suggests to you, and how your social media feeds are presented. It powers autonomous vehicles and machines that can diagnose medical conditions based on images.

When companies today deploy artificial intelligence programs, they are most likely using machine learning — so much so that the terms are often used interchangeably, and sometimes ambiguously. Machine learning is a subfield of artificial intelligence that gives computers the ability to learn without explicitly being programmed.

"In just the last five or 10 years, machine learning has become a critical way, arguably the most important way, most parts of AI are done," said MIT Sloan professorThomas W. Malone, the founding director of the MIT Center for Collective Intelligence. "So that's why some people use the terms AI and machine learning almost as synonymous … most of the current advances in AI have involved machine learning."

With the growing ubiquity of machine learning, everyone in business is likely to encounter it and will need some working knowledge about this field. A 2020 Deloitte survey found that 67% of companies are using machine learning, and 97% are using or planning to use it in the next year.

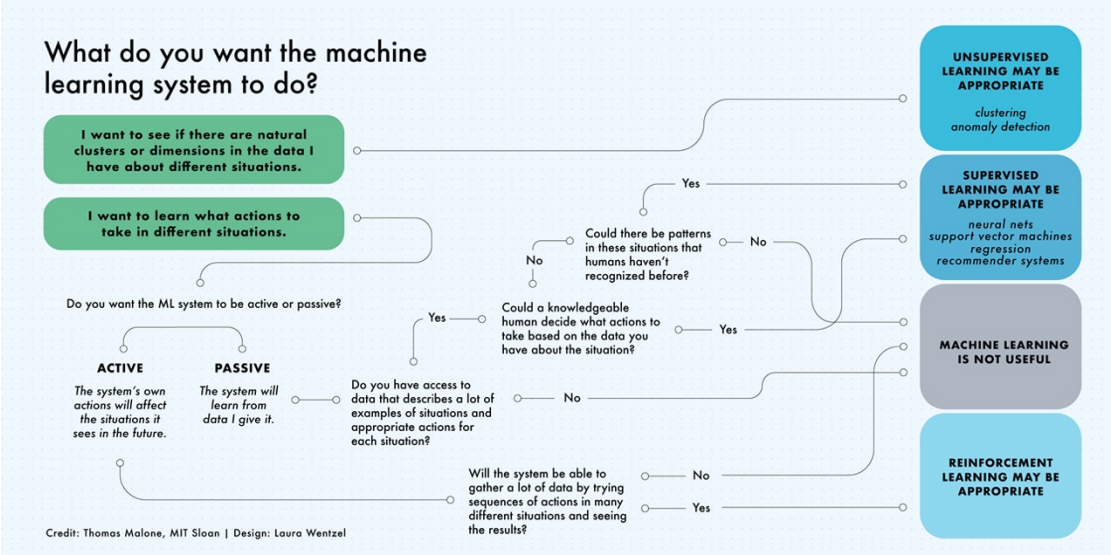

Figure 2-1 Source: Thomas Malone | MIT Sloan [5]

From manufacturing to retail and banking to bakeries, even legacy companies are using machine learning to unlock new value or boost efficiency. "Machine learning is changing, or

will change, every industry, and leaders need to understand the basic principles, the potential, and the limitations," said MIT computer science professor Aleksander Madry, director of the MIT Center for Deployable Machine Learning.

While not everyone needs to know the technical details, they should understand what the technology does and what it can and cannot do, Madry added. "I don't think anyone can afford not to be aware of what's happening."

That includes being aware of the social, societal, and ethical implications of machine learning. "It's important to engage and begin to understand these tools, and then think about how you're going to use them well. We have to use these [tools] for the good of everybody," said Dr. Joan LaRovere, MBA '16, a pediatric cardiac intensive care physician and co-founder of the nonprofit The Virtue Foundation. "AI has so much potential to do good, and we need to really keep that in our lenses as we're thinking about this. How do we use this to do good and better the world?

2.2.2 What is machine learning?

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

The goal of AI is to create computer models that exhibit "intelligent behaviors" like humans, according to Boris Katz, a principal research scientist and head of the InfoLab Group at CSAIL. This means machines that can recognize a visual scene, understand a text written in natural language, or perform an action in the physical world.

Machine learning is one way to use AI. It was defined in the 1950s by AI pioneer Arthur Samuel as "the field of study that gives computers the ability to learn without explicitly being programmed."

The definition holds true, according toMikey Shulman, a lecturer at MIT Sloan and head of machine learning at Kensho, which specializes in artificial intelligence for the finance and U.S. intelligence communities. He compared the traditional way of programming computers, or "software 1.0," to baking, where a recipe calls for precise amounts of ingredients and tells the baker to mix for an exact amount of time. Traditional programming similarly requires creating detailed instructions for the computer to follow. [16]

But in some cases, writing a program for the machine to follow is time-consuming or impossible, such as training a computer to recognize pictures of different people. While humans can do this task easily, it's difficult to tell a computer how to do it. Machine learning takes the approach of letting computers learn to program themselves through experience.

Machine learning starts with data — numbers, photos, or text, like bank transactions, pictures of people or even bakery items, repair records, time series data from sensors, or sales reports. The data is gathered and prepared to be used as training data, or the information the machine learning model will be trained on. The more data, the better the program.

From there, programmers choose a machine learning model to use, supply the data, and let the computer model train itself to find patterns or make predictions. Over time the human programmer can also tweak the model, including changing its parameters, to help push it toward more accurate results. (Research scientist Janelle Shane's website AI Weirdness is an entertaining look at how machine learning algorithms learn and how they can get things wrong — as happened when an algorithm tried to generate recipes and created Chocolate Chicken Chicken Cake.)

Some data is held out from the training data to be used as evaluation data, which tests how accurate the machine learning model is when it is shown new data. The result is a model that can be used in the future with different sets of data.

Successful machine learning algorithms can do different things, Malone wrote in a recent research brief about AI and the future of work that was co-authored by MIT professor and CSAIL director Daniela Rus and Robert Laubacher, the associate director of the MIT Center for Collective Intelligence

"The function of a machine learning system can be descriptive, meaning that the system uses the data to explain what happened; predictive, meaning the system uses the data to predict what will happen; or prescriptive, meaning the system will use the data to make suggestions about what action to take," the researchers wrote.

There are three subcategories of machine learning:

- Supervised machine learning models are trained with labeled data sets, which allow the models to learn and grow more accurate over time. For example, an algorithm would be trained with pictures of dogs and other things, all labeled by humans, and the machine would learn ways to identify pictures of dogs on its own. Supervised machine learning is the most common type used today.

In unsupervised machine learning, a program looks for patterns in unlabeled data. Unsupervised machine learning can find patterns or trends that people aren't explicitly looking for. For example, an unsupervised machine learning program could look through online sales data and identify different types of clients making purchases.

Reinforcement machine learning trains machines through trial and error to take the best action by establishing a reward system. Reinforcement learning can train models to play games or

train autonomous vehicles to drive by telling the machine when it made the right decisions, which helps it learn over time what actions it should take.

In the Work of the Future brief, Malone noted that machine learning is best suited for situations with lots of data — thousands or millions of examples, like recordings from previous conversations with customers, sensor logs from machines, or ATM transactions. For example, Google Translate was possible because it "trained" on the vast amount of information on the web, in different languages.

In some cases, machine learning can gain insight or automate decision-making in cases where humans would not be able to, Madry said. "It may not only be more efficient and less costly to have an algorithm do this, but sometimes humans just literally are not able to do it," he said.

Google search is an example of something that humans can do, but never at the scale and speed at which the Google models are able to show potential answers every time a person types in a query, Malone said. "That's not an example of computers putting people out of work. It's an example of computers doing things that would not have been remotely economically feasible if they had to be done by humans."

Machine learning is also associated with several other artificial intelligence subfields

- Natural language processing is a field of machine learning in which machines learn to understand natural language as spoken and written by humans, instead of the data and numbers normally used to program computers. This allows machines to recognize language, understand it, and respond to it, as well as create new text and translate between languages. Natural language processing enables familiar technology like chatbots and digital assistants like Siri or Alexa.
- Neural networks are a commonly used, specific class of machine learning algorithms. Artificial neural networks are modeled on the human brain, in which thousands or millions of processing nodes are interconnected and organized into layers. In an artificial neural network, cells, or nodes, are connected, with each cell processing inputs and producing an output that is sent to other neurons. Labeled data moves through the nodes, or cells, with each cell performing a different function. In a neural network trained to identify whether a picture contains a cat or not, the different nodes would assess the information and arrive at an output that indicates whether a picture features a cat.
- Deep learning networks are neural networks with many layers. The layered network can process extensive amounts of data and determine the "weight" of each link in the network — for example, in an image recognition system, some layers of the neural network might detect individual features of a face, like eyes, nose, or mouth, while

another layer would be able to tell whether those features appear in a way that indicates a face.

Like neural networks, deep learning is modeled on the way the human brain works and powers many machine learning uses, like autonomous vehicles, chatbots, and medical diagnostics.

“The more layers you have, the more potential you have for doing complex things well,” Malone said. Deep learning requires a great deal of computing power, which raises concerns about its economic and environmental sustainability.

3

Theoretical background

3.1 Bidirectional Encoder Representation from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

3.1.1 Background

In the field of computer vision, researchers have repeatedly shown the value of transfer learning — pre-training a neural network model on a known task, for instance ImageNet, and then performing fine-tuning — using the trained neural network as the basis of a new purpose-specific model. In recent years, researchers have been showing that a similar technique can be useful in many natural language tasks.

A different approach, which is also popular in NLP tasks and exemplified in the recent ELMo paper, is feature-based training. In this approach, a pre-trained neural network produces word embeddings which are then used as features in NLP models.

3.1.2 How BERT works

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The chart below is a high-level description of the Transformer encoder. The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

When training language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence (e.g. "The child came home from ___"), a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies:

3.1.3 Masked LM (MLM)

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

- Adding a classification layer on top of the encoder output.
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
- Calculating the probability of each word in the vocabulary with softmax

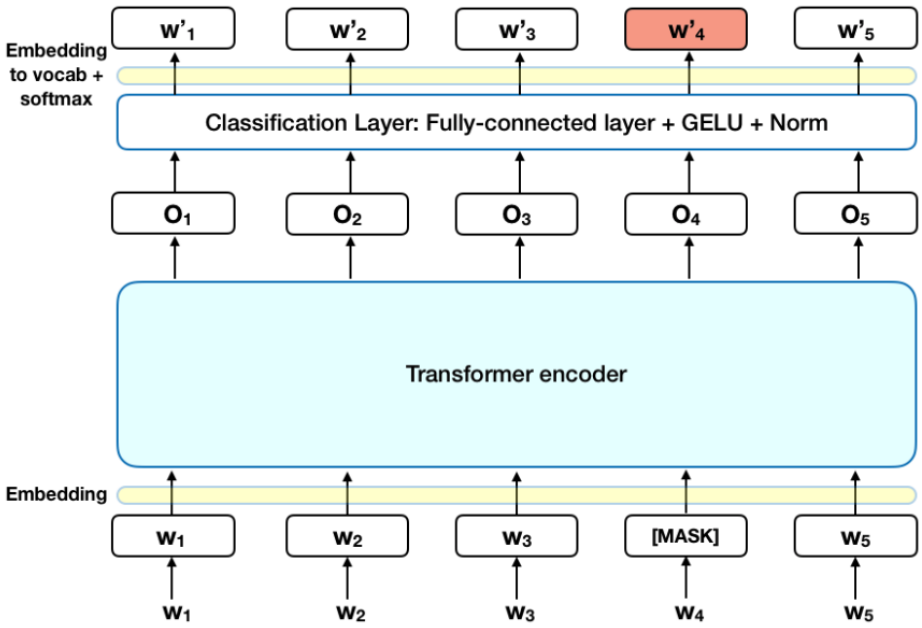

Figure 3-1 Architecture of Bidirectional Encoder Representations from Transformers (BERT) [6]

The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words. As a consequence, the model converges slower than directional models, a characteristic which is offset by its increased context awareness (see Takeaways #3).

Note: In practice, the BERT implementation is slightly more elaborate and doesn't replace all of the 15% masked words. See Above for more details and additional info

3.1.4 Word Masking

Training the language model in BERT is done by predicting 15% of the tokens in the input, that were randomly picked. These tokens are pre-processed as follows — 80% are replaced with a "[MASK]" token, 10% with a random word, and 10% use the original word. The intuition that led the authors to pick this approach is as follows (Thanks to Jacob Devlin from Google for the insight):

If we used [MASK] 100% of the time the model wouldn't necessarily produce good token representations for non-masked words. The non-masked tokens were still used for context, but the model was optimized for predicting masked words.

3.1.5 Next Sentence Prediction (NSP)

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent

sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

- A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
- A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
- A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

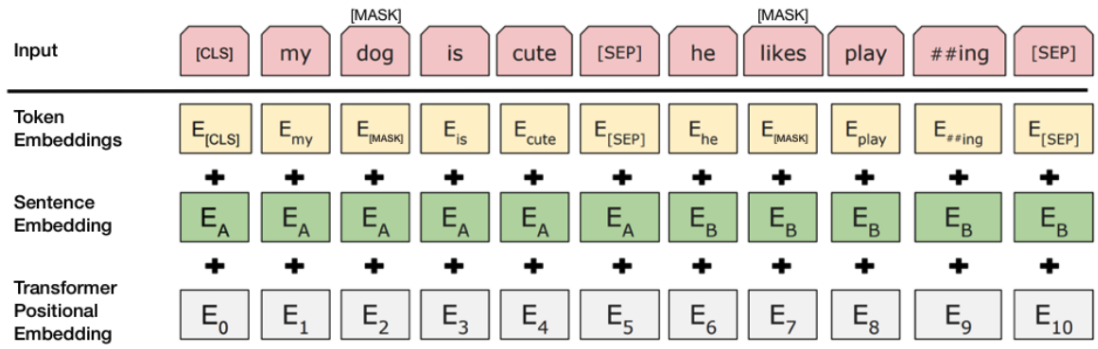

Figure 3-2 BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. [1]

To predict if the second sentence is indeed connected to the first, the following steps are performed:

- The entire input sequence goes through the Transformer model.
- The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
- Calculating the probability of IsNextSequence with softmax.

When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

3.1.6 How to use BERT (Fine-tuning)

Using BERT for a specific task is relatively straightforward:

BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:

- Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
- In Question Answering tasks (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer.
- In Named Entity Recognition (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label.

In the fine-tuning training, most hyper-parameters stay the same as in BERT training, and the paper gives specific guidance (Section 3.5) on the hyper-parameters that require tuning. The BERT team has used this technique to achieve state-of-the-art results on a wide variety of challenging natural language tasks, detailed in Section 4 of the paper.

3.1.7 Different Variations of BERT

3.1.7.1 BERT Large vs BERT Base

Both models are following the same approach but they have some differences. As the number of layers is BERT large is increased so does the number of parameters (weights) and number of attention heads increases. BERT is based on stacked layers of encoders. The difference between BERT base and BERT large is on the number of encoder layers. BERT base model has 12 encoder layers stacked on top of each other whereas BERT large has 24 layers of encoders stacked on top of each other. BERT base has a total of 12 attention heads and 110 million parameters. Whereas BERT large has 16 attention heads with 340 million parameters. BERT base has 768 hidden layers whereas BERT large has 1024 hidden layers. As we can see in the image above, the results of GLUE benchmarks show us that the BERT performs better than the other models. And BERT large increases the performance of BERT base further.

The image below shows SWAG Dev and test accuracy. BERT beats other models where BERT large performs better than BERT base.

3.1.7.2 DistilBERT

DistilBERT is a transformers model and the main advantage is that's smaller and faster than the Base or Large BERT variations. It is also pre-trained but in a slightly different way with almost half the number of parameters [5].

This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts using the BERT base model. More precisely, it was pretrained with three objectives:

- Distillation loss: The training goal was to be as close as possible and has the same probabilities as the BERT base model.
- Masked language modeling (MLM): this is part of the original training loss of the BERT base model. When taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.
- Cosine embedding loss: the model was also trained to generate hidden states as close as possible as the BERT base model.

This way, the model learns the same portion of the English language than its teacher model, while being faster for inference or downstream tasks. More specifically the size could be reduced by 40% while retain almost the same level of language understanding (97%) [5].

In the below figure you can find some metrics where we can see that the differences compared to the performance gain and the size of the model is not so big.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Figure 3-3 Performance comparison of DistilBERT and BERT-base [5]

3.1.8 Takeaways

- Model size matters, even at huge scale. BERT_large, with 345 million parameters, is the largest model of its kind. It is demonstrably superior on small-scale tasks to BERT_base, which uses the same architecture with "only" 110 million parameters.

- With enough training data, more training steps == higher accuracy. For instance, on the MNLI task, the BERT_base accuracy improves by 1.0% when trained on 1M steps (128,000 words batch size) compared to 500K steps with the same batch size.
- BERT's bidirectional approach (MLM) converges slower than left-to-right approaches (because only 15% of words are predicted in each batch) but bidirectional training still outperforms left-to-right training after a small number of pre-training steps.

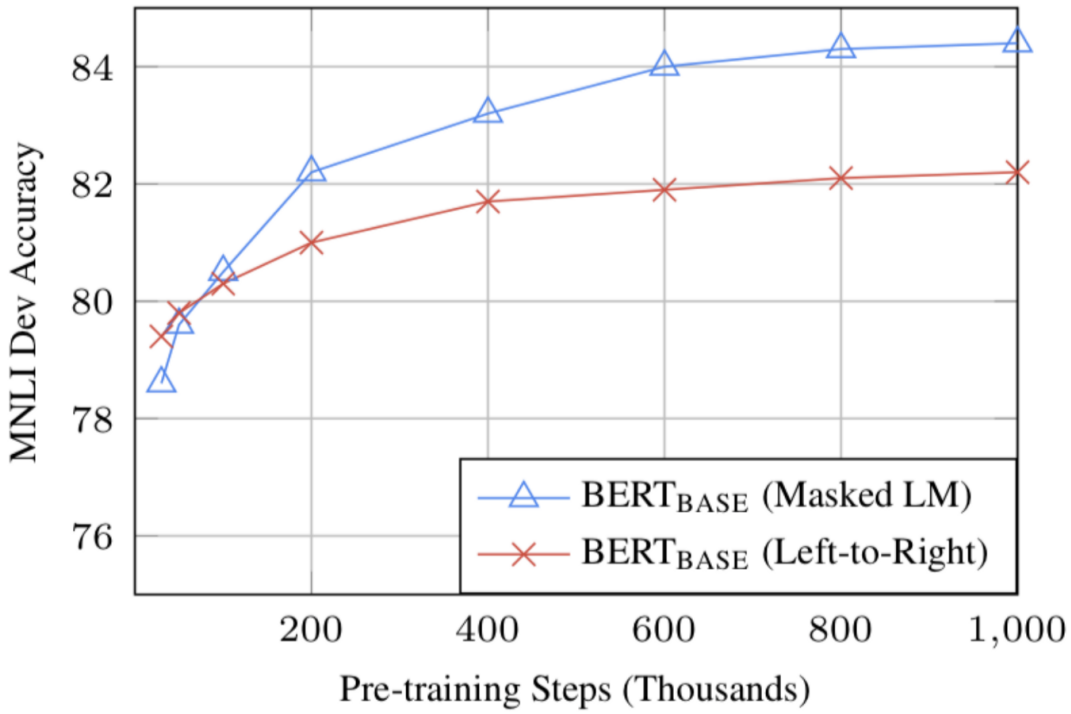

Figure 3-4 Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k. [1]

3.1.9 Conclusion

BERT is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. The fact that it's approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future. In this summary, we attempted to describe the main ideas of the paper while not drowning in excessive technical details. For those wishing for a deeper dive, we highly recommend reading the full article and ancillary articles referenced in it. Another useful reference is the BERT source code and models, which cover 103 languages and were generously released as open source by the research team.

4

High level overview of the Web Application Logic

The main challenge of this thesis is to integrate the web application with the BERT model and give the user the opportunity to summarize their texts. In order to achieve this task, we created a web UI with a text form that will pass the original text to the model and then after processing the original text and run the overall model as described in previous sections, we will return a new summary based on the original text to the user. The overall technical implementation will be described with more details in the next chapter with code snippets from our application. Below are the steps we follow in order to accomplish this task.

4.1 Web Application

The web application consists by the front-end and the back-end implementation

4.1.1 Front-end

The front-end is a simple web form where the user can submit the summarized text and also decide if he wants to parametrize some other variables as an input to the backend model.

The available parameters are:

- Model: The user is able to choose between three different models, for more details about the differences you can check chapter 3:
    - BERT Large
    - BERT Base
    - DistillBERT
- Number of Sentences: The user can configure the number of sentences that needs to be returned for the final summarized text
- Min Length: The minimum length of a sentence that will be used as an input to the model.

- Max Length: The maximum length of a sentence that will be used as an input to the model.

4.1.2 Back-end

In the back-end we have some function in order to start the overall automation process of text summarization. Below are the most important parts:

- Form definition: we define the form that will be created in the front-end during the execution of the application.
- Route functions: In the route function we have all the logic for the process of the data form the front-end.
- Model: we prepare the text we received in order to submit to the final model
- Bert-summarizer: a Python class that has all the logic of the summarization process we describe in the next section.

4.2 Summarization Process

- Sentence Tokenization: The first step is to tokenize the text into different sentences in order to give it as an input to the BERT model. Sentence tokenization is the process of splitting text into individual sentences.
- Produce sentence embeddings: After the tokenization of our initial text, we need to give the sentences as an input to the BERT model in order to produce sentence embeddings. Embeddings are representation of sentences as vectors.
- Cluster Embeddings: as next step we create clusters by using k-means for the embeddings in order to leverage the information for the next steps
- Select n-Embedding: to return the final sentences and construct the summarized text we return the n-embeddings that are closer to the centroid of the clusters.
- Construct the summarized text: then we concatenate all the sentences and prepare the final text.

In the next chapter we will go through the overall process we described and different components with more examples and code snippets.

5

The Solution

As we briefly described the overall process in the previous section, in this section we will go into more details about the implementation of the problem. More examples could be found in the Chapter 7 along with the tools, languages, frameworks and libraries we used.

5.1 The Application

5.1.1 Project Structure

In the below picture you can find the basic folder project structure, the main folder of the project called mythesis is the root folder and has the virtual environment create by pip (more details in the chapter 7), the basic configuration and requirements.txt file and the app directory which is the main directory for our flask application. The app directory contains one subdirectory called "templates" where we store our HTML templates and other application files that we will describe in this chapter.

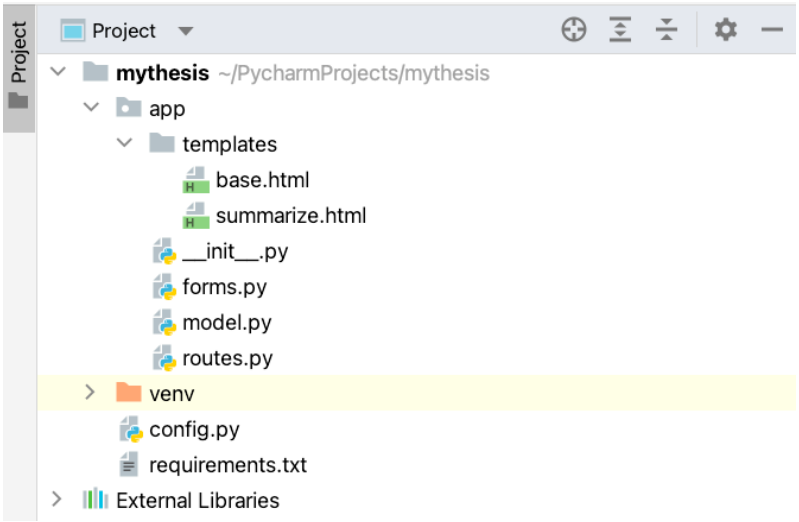

Figure 5-1 Project Structure

5.1.1.1 Routes

In the routes.py file we have all the logic for our app routing. With app routing we can define the way we handle the logic for different URLs that our app could support. In order to define different routes for different URLs we use the "@" decorator before the function definition. In Flask you can have multiple routes handled by the same function by defining multiple decorators just above the function. Below is a snippet of the function

```
@app.route('/', methods=['GET', 'POST'])
@app.route('/summarize', methods=['GET', 'POST'])
@app.route('/index', methods=['GET', 'POST'])
def summarize():
    form = SummarizeForm()
    if form.validate_on_submit():
        initial_word_count = str(len(form.text.data.split()))
        my_model = form.model.data
        my_sentences = form.sentences.data
        min_length = form.min_length.data
        max_length = form.max_length.data
        start = time.time()
        returned_text = runit(form.text.data,
                              sentences=my_sentences,
                              model=my_model,
                              min_length=min_length,
                              max_length=max_length)
        final_word_count = str(len(returned_text.split()))
        form.sum_text.data = returned_text
        percentage = str(
            (round(100-(100*int(final_word_count)/
                        int(initial_word_count)))))
        stop = time.time()
        my_time = round(stop - start, 2)
        flash(f'Your Summary is ready!', 'success')
        flash(f'Initial Word Number: ' + initial_word_count, 'info')
        flash(f'Final Word Number: ' + final_word_count, 'info')
        flash(f'Reduction: ' + percentage + '%', 'info')
        flash(f'Model: ' + str(my_model), 'info')
        flash(f'No Sentences: ' + str(my_sentences), 'info')
        flash(f'Min Length: ' + str(min_length), 'info')
        flash(f'Max Length: ' + str(max_length), 'info')
        flash(f'Total time: ' + str(my_time) + 's', 'alert')
        return render_template('summarize.html',
                               title='Text Summarization',
                               text=returned_text, form=form,
                               final=final_word_count,
                               initial=initial_word_count)
    return render_template('summarize.html',
                           title='Text Summarization', form=form)
```

Figure 5-2 Routes.py

As you can see there are three decorators that are related to different URLs of our web app. When the user prompts the "/index" as an example the function will run and start all the logic.

As you can see the first call is to create a form by using the function "SummarizeForm". In the next section we will describe the forms.py and how we create the form.

5.1.1.2 Form

As we described in the previous section when a user requests a specific URL a function runs in the backend and do the necessary logic. In this section we will describe the definition of the form as part of the WTF forms library of Flask.

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, SelectField
from wtforms.validators import DataRequired, Length
from wtforms.widgets import TextArea
from wtforms.fields.html5 import IntegerField

class SummarizeForm(FlaskForm):
    text = StringField('Submit your text here', widget=TextArea(),
                       render_kw={'rows': 15},
                       validators=[DataRequired(),
                                   Length(max=100000)])
    sum_text = StringField('Summarized Text',
                           widget=TextArea(),
                           render_kw={'readonly': True,
                                      'rows': 8},
                           validators=[Length(max=100000)])
    model = SelectField('Choose the model',
                        choices=[(state, state) for state in
                                 ('bert-base-uncased',
                                  'distilbert-base-uncased',
                                  'bert-large-uncased')])
    sentences = IntegerField('Number of Sentencies',
                             render_kw={'min': 3,
                                        'default': 3,
                                        'value': 3})
    min_length = IntegerField('Min Length',
                              render_kw={'min': 10,
                                         'default': 25,
                                         'value': 25})
    max_length = IntegerField('Max Length',
                              render_kw={'min': 150,
                                         'default': 500,
                                         'value': 500})
    submit = SubmitField('Get Summary',
                         render_kw={'onclick': 'loading()'})
```

Figure 5-3 forms.py

With the usage of flask_wrf we imported the FlaskForm class and we defined the form that is presented to the user. The result of the above is the following screen from our web application which is the UI.

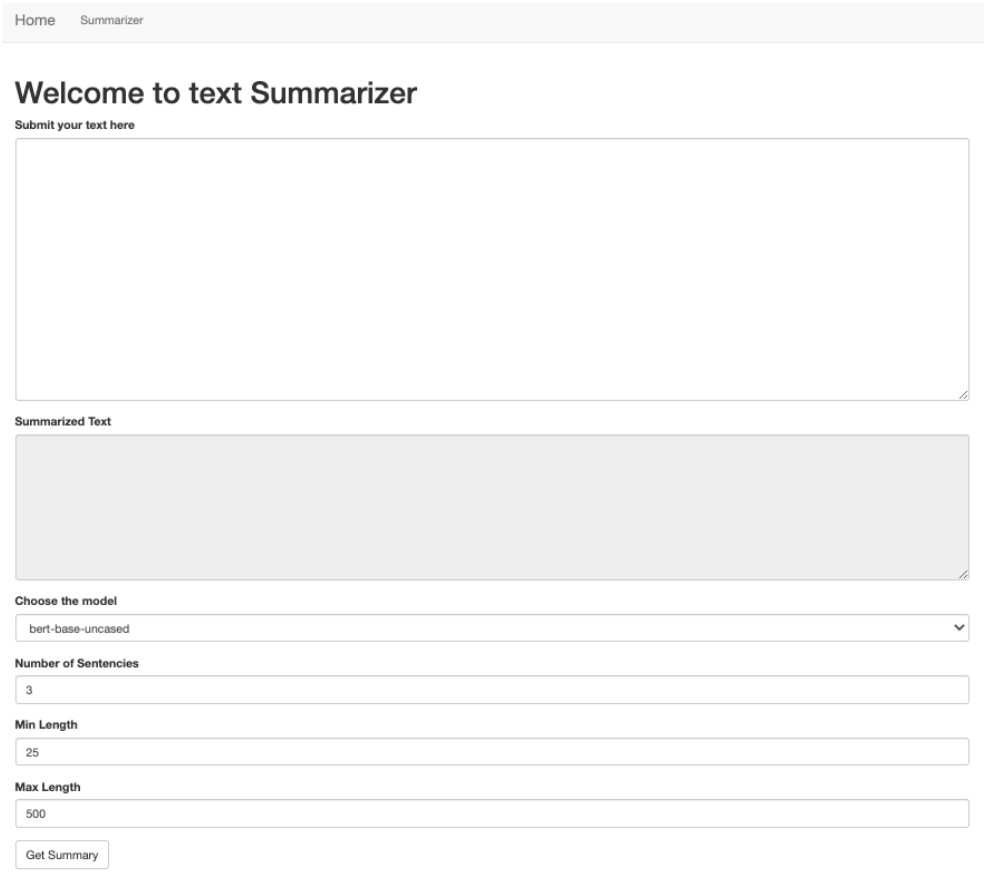

Figure 5-4 Web UI

All the options for the users are available in order to configure some options as we mentioned in Chapter 4. When the user requests the /index URL the Flask web server runs the function inside the routes.py and then renders the template and pass the form as an argument. Chek the snippet of code in the below picture.

5.1.1.3 Model

```python
import nltk
import ssl
import re
from summarizer import Summarizer


class TextModel(object):

    def __init__(self):
        self.init_checks()

    @staticmethod
    def run_summarizer(doc, num_sentences=4, min_length=25,
                       max_length=500,
                       model='distilbert-base-uncased'):
        doc = re.sub(r'\n|\r', ' ', doc)
        doc = re.sub(r' +', ' ', doc)
        doc = doc.strip()

        sm = Summarizer(model=model)
        result = sm(body=doc, num_sentences=num_sentences,
                    min_length=min_length,
                    max_length=max_length)

        result = ' '.join(nltk.sent_tokenize(result))
        return result

```

Figure 5-5 model.py

The main model that will be called by our routes.py function and pass the data from the user to the model could be found in the above picture. As you can see we leverage the Summarizer class that will describe in the next section and send all the data defined by the user.

5.1.1.4 Base HTML Template

The code in the picture of HTML Base Template is the HTML code related to a base template that consists of the navigation bar and will be used as a parent for all our HTML pages. The idea is to extend a base template that it is built by bootstrap framework called "bootstrap/base.html".

```
{% extends 'bootstrap/base.html' %}
< !doctype html >
{% block title %}
      {% if title %}
      {{ title }} - Extractive Text Summarization{% else %} Welcome to Extractive Text Summarization{% endif %}
{% endblock %}
{% block navbar %}
    <nav class="navbar navbar-default">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
                        data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="{{ url_for('summarize') }}">Home</a>
            </div>
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav">
                    <li><a href="{{ url_for('summarize') }}">Summarizer</a></li>
                </ul>
            </div>
        </div>
    </nav>
{% endblock %}
```

Figure 5-6 HTML Base Template

This template is using Jinja as well (check Chapter 7 for details around Jinja) in order to use python like expression inside the HTML template.

5.1.1.5 Summarize HTML Template

```
{% extends "base.html" %}
{% import 'bootstrap/wtf.html' as wtf %}

{% block app_content %}
    <h1>Welcome to text Summarizer
        <img id="loader" style="display:none"
             src="https://c.tenor.com/I6kN-6X7nhAAAAAi/loading-buffering.gif"
             alt="image" width="50" height="50"> </h1>
    <div class="row">
        <div class="col-md-12">
            {{ wtf.quick_form(form) }}
      </div>
    </div>
    <script type="text/javascript">
        function loading(){
            document.getElementById("loader").style.display = 'inline';
        }
    </script>

{% endblock %}

```

Figure 5-7 Summarize HTML Template

In the summarize html template we extend our base HTML and also, we refer inside the jinja2 statement block we utilize the form we created as defined in the forms.py picture above. The result is the web UI that is available to the user.

5.1.2 Text Summarizer

The below code is fork of the code of the Python module "bert-extractive-summarizer" [15].

We will take a look in the class that is used in our model in the previous section.

```
307  class Summarizer(ModelProcessor):
308
309      def __init__(
310          self,
311          model: str = 'bert-large-uncased',
312          custom_model: PreTrainedModel = None,
313          custom_tokenizer: PreTrainedTokenizer = None,
314          hidden: Union[List[int], int] = -2,
315          reduce_option: str = 'mean',
316          sentence_handler: SentenceHandler = SentenceHandler(),
317          random_state: int = 12345,
318          hidden_concat: bool = False
319      ):
```

Figure 5-8 Summarizer Class [15]

The below summarizer class uses a pretrained model that completes the below steps:

- Embeds the sentences
- Runs a clustering algorithm
- Finds the sentences that are closest to the cluster's centroids.
- Returns the sentences

Before we create summaries of the text the we tokenize the incoming text into clean sentences then we pass the tokenized sentences to the BERT model for inference to output embeddings, and then cluster the embeddings with K-Means, selecting the embedded sentences that were closest to the centroid as the candidate summary sentences and reconstruct the text at the end.

5.1.2.1 Text Tokenization

Before submit the text into the model of the library we tokenize it into different sentences by using the NLTK tokenization function. The sentences are input to the BERT model in order to create the embeddings as described in the next section.

5.1.2.2 Text Embeddings

Compared to performance of other NLP algorithms on sentence embedding as we describe in the next Chapter 6 the BERT architecture was selected. BERT builds on top of the transformer architecture, but its objectives are specific for pre-training. On one step, it randomly masks out 10% to 15% of the words in the training data, attempting to predict the masked words, and the other step takes in an input sentence and a candidate sentence, predicting whether the candidate sentence properly follows the input sentence (Devlin, Chang, Lee, & Toutanova, 2018). [1] This process can take several days to train, even with a substantial number of GPUs. Due to this fact, Google released two BERT models for public consumption, where one had 110 million parameters and the other contained 340 million parameters (Devlin, Chang, Lee, & Toutanova, 2018). Using the default pre-trained BERT model, one can select multiple layers for embeddings. Using the classification (cls) layer of BERT produces the necessary N x E matrix for clustering,

where N is the number of sentences and E is the embeddings dimension, but the output of the [cls] layer does not necessarily produce the best embedding representation for sentences. Due to the nature of the BERT architecture, outputs for other layers in the network produced N x W x E Embeddings where W equaled the tokenized words. To get around this issue, the embeddings can be averaged or maxed to produce an N x E matrix. After experiments with Udacity extractive summarizations on Udacity lectures, it was determined that the second to last averaged layer (N-2) produced the best embeddings for representations of words [1]. By using this class we can create our own custom model with different number of layers but that was not in scope for the completion of this Thesis due to resources limitations and huge amount of time to train the BERT model in such a huge dataset.

5.1.2.3 Clustering Embeddings

The bert-summarizer class finally, once the N-2-layer embeddings were completed, the

N x E matrix was ready for clustering. In the class definition we enhanced the parameter K, which would represent the number of clusters and requested sentences for the final summary output. K-Means selected for clustering incoming embeddings from the BERT model. From the clusters, the sentences closest to the centroids were selected or the final summary.

6

Evaluation

For the evaluation of our solution, we referred to the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" that has an extensive comparison of our basic models. Apart from that we run some manual or human experimentation where we used some small texts from the web and then compared the summary of the document with the original one to understand the quality of the outcome.

6.1 Choose a Model

"The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018a) is a collection of diverse natural language understanding tasks. Here we describe in detail the parameters whose values we will measure and explain why we chose these parameters" [1].

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Figure 6-1 GLUE test results [1]

In the above table it's obvious that the BERT model outperforms the rest of the compared models in almost all categories. During the initial research we decided to choose the model based on the above table. In the next section we will describe the different parameters of the GLUE Evaluation model that presented in the above table.

6.2 GLUE as Evaluation system

GLUE stands for the General Language Understanding Evaluation benchmark. It could be used to evaluate Machine Learning models in different categories. There are other evaluation systems like SquAD that stands as the Standford Question Answering Dataset.

For the GLUE model there are different tasks we will describe below.

6.2.1 Single Sentence Tasks

- CoLA The Corpus of Linguistic Acceptability (Warstadt et al., 2018) consists of English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence.
- SST-2 The Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence. [14]

6.2.2 Inference Tasks

- MLNI: The Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a crowdsourced collection of sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral).
- QNLI The Stanford Question Answering Dataset (Rajpurkar et al. 2016) is a question-answering dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph (drawn from Wikipedia) contains the answer to the corresponding question (written by an annotator).
- WNLI The Winograd Schema Challenge (Levesque et al., 2011) is a reading comprehension task in which a system must read a sentence with a pronoun and select the referent of that pronoun from a list of choices. [14]

6.3 Manual Tests

In this section we will give some examples of text we summarized and manual evaluated the quality of those for your own reference:

6.3.1 Example 1

Below you can find the results along with the parameters:

6.3.1.1 Parameters and Stats

- Initial Word Number: 736
- Final Word Number: 458
- Reduction: 38%
- Model: bert-base-uncased
- No Sentences: 20
- Min Length: 25
- Max Length: 500
- Total time: 15.93s

6.3.1.2 Initial Text

“Barack Hussein Obama II (/bəˈrɑːk huːˈseɪn oʊˈbɑːmə/ (About this soundlisten) bə-RAHK hoo-SAYN oh-BAH-mə;[1] born August 4, 1961) is an American politician and retired attorney who served as the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, Obama was the first African-American president of the United States. He previously served as a U.S. senator from Illinois from 2005 to 2008 and as an Illinois state senator from 1997 to 2004.

Obama was born in Honolulu, Hawaii. After graduating from Columbia University in 1983, he worked as a community organizer in Chicago. In 1988, he enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. After graduating, he became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. Turning to elective politics, he represented the 13th district in the Illinois Senate from 1997 until 2004, when he ran for the U.S. Senate. Obama received national attention in 2004 with his March Senate primary win, his well-received July Democratic National Convention keynote address, and his landslide November election to the Senate. In 2008, a year after beginning his campaign, and after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama was elected over Republican nominee John McCain in the general election and was inaugurated alongside his running mate, Joe Biden, on January 20, 2009. Nine months later, he was named the 2009 Nobel Peace Prize laureate.

Obama signed many landmark bills into law during his first two years in office. The main reforms include: the Affordable Care Act (ACA or "Obamacare"), although without a public health insurance option; the Dodd–Frank Wall Street Reform and Consumer Protection Act;

and the Don't Ask, Don't Tell Repeal Act of 2010. The American Recovery and Reinvestment Act of 2009 and Tax Relief, Unemployment Insurance Reauthorization, and Job Creation Act of 2010 served as economic stimuli amidst the Great Recession. After a lengthy debate over the national debt limit, he signed the Budget Control and the American Taxpayer Relief Acts. In foreign policy, he increased U.S. troop levels in Afghanistan, reduced nuclear weapons with the United States–Russia New START treaty, and ended military involvement in the Iraq War. He ordered military involvement in Libya for the implementation of the UN Security Council Resolution 1973, contributing to the overthrow of Muammar Gaddafi. He also ordered the military operation that resulted in the killing of Osama bin Laden.

After winning re-election by defeating Republican opponent Mitt Romney, Obama was sworn in for a second term in 2013. During this term, he promoted inclusion for LGBT Americans. His administration filed briefs that urged the Supreme Court to strike down same-sex marriage bans as unconstitutional (United States v. Windsor and Obergefell v. Hodges); same-sex marriage was legalized nationwide in 2015 after the Court ruled so in Obergefell. He advocated for gun control in response to the Sandy Hook Elementary School shooting, indicating support for a ban on assault weapons, and issued wide-ranging executive actions concerning global warming and immigration. In foreign policy, he ordered military interventions in Iraq and Syria in response to gains made by ISIL after the 2011 withdrawal from Iraq, promoted discussions that led to the 2015 Paris Agreement on global climate change, oversaw and ultimately apologized for the deadly Kunduz hospital airstrike, continued the process of ending U.S. combat operations in Afghanistan in 2016, initiated sanctions against Russia following the invasion in Ukraine and again after interference in the 2016 U.S. elections, brokered the Joint Comprehensive Plan of Action nuclear deal with Iran, and normalized U.S. relations with Cuba. Obama nominated three justices to the Supreme Court: Sonia Sotomayor and Elena Kagan were confirmed as justices, while Merrick Garland faced partisan obstruction from the Republican-majority Senate led by Mitch McConnell, which never held hearings or a vote on the nomination. Obama left office in January 2017 and continues to reside in Washington, D.C.[2][3

During Obama's terms as president, the United States' reputation abroad, as well as the American economy, significantly improved.[4] Obama's presidency has generally been regarded favorably, and evaluations of his presidency among historians, political scientists, and the general public frequently place him among the upper tier of American presidents. Since leaving office, Obama has remained active in Democratic politics, including appearing at the 2020 Democratic National Convention."

6.3.1.3 Summarized Text

“Barack Hussein Obama II (/bəˈrɑːk huːˈseɪn oʊˈbɑːmə/ (About this soundlisten) bə-RAHK hoo-SAYN oh-BAH-mə;[1] born August 4, 1961) is an American politician and retired attorney who served as the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, Obama was the first African-American president of the United States. He previously served as a U.S. senator from Illinois from 2005 to 2008 and as an Illinois state senator from 1997 to 2004. Obama was born in Honolulu, Hawaii. After graduating from Columbia University in 1983, he worked as a community organizer in Chicago. In 1988, he enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. In 2008, a year after beginning his campaign, and after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama was elected over Republican nominee John McCain in the general election and was inaugurated alongside his running mate, Joe Biden, on January 20, 2009. Nine months later, he was named the 2009 Nobel Peace Prize laureate. Obama signed many landmark bills into law during his first two years in office. The American Recovery and Reinvestment Act of 2009 and Tax Relief, Unemployment Insurance Reauthorization, and Job Creation Act of 2010 served as economic stimuli amidst the Great Recession. After a lengthy debate over the national debt limit, he signed the Budget Control and the American Taxpayer Relief Acts. In foreign policy, he increased U.S. troop levels in Afghanistan, reduced nuclear weapons with the United States–Russia New START treaty, and ended military involvement in the Iraq War. He ordered military involvement in Libya for the implementation of the UN Security Council Resolution 1973, contributing to the overthrow of Muammar Gaddafi. He also ordered the military operation that resulted in the killing of Osama bin Laden. After winning re-election by defeating Republican opponent Mitt Romney, Obama was sworn in for a second term in 2013. During this term, he promoted inclusion for LGBT Americans. Obama nominated three justices to the Supreme Court: Sonia Sotomayor and Elena Kagan were confirmed as justices, while Merrick Garland faced partisan obstruction from the Republican-majority Senate led by Mitch McConnell, which never held hearings or a vote on the nomination. Obama left office in January 2017 and continues to reside in Washington, D.C.[2][3] During Obama's terms as president, the United States' reputation abroad, as well as the American economy, significantly improved. [4] Obama's presidency has generally been regarded favorably, and evaluations of his presidency among historians, political scientists, and the general public frequently place him among the upper tier of American presidents. Since leaving office, Obama has remained active in Democratic politics, including appearing at the 2020 Democratic National Convention.”

6.3.2 Example 2

6.3.2.1 Parameters and Stats

- Initial Word Number: 525
- Final Word Number: 88
- Reduction: 83%
- Model: bert-large-uncased
- No Sentences: 3
- Min Length: 25
- Max Length: 500
- Total time: 26.14s

6.3.2.2 Initial Text

“Greece (Greek: Ελλάδα, romanized: Elláda, [eˈlaða]), officially the Hellenic Republic,[b] is a country located in Southeast Europe. Its population is approximately 10.7 million as of 2018; Athens is its largest and capital city, followed by Thessaloniki. Situated on the southern tip of the Balkans, Greece is located at the crossroads of Europe, Asia, and Africa. It shares land borders with Albania to the northwest, North Macedonia and Bulgaria to the north, and Turkey to the northeast. The Aegean Sea lies to the east of the mainland, the Ionian Sea to the west, the Cretan Sea and the Mediterranean Sea to the south. Greece has the longest coastline on the Mediterranean Basin and the 11th longest coastline in the world at 13,676 km (8,498 mi) in length, featuring many islands, of which 227 are inhabited. Eighty percent of Greece is mountainous, with Mount Olympus being the highest peak at 2,918 metres (9,573 ft). The country consists of nine traditional geographic regions: Macedonia, Central Greece, the Peloponnese, Thessaly, Epirus, the Aegean Islands (including the Dodecanese and Cyclades), Thrace, Crete, and the Ionian Islands.

Greece is considered the cradle of Western civilization, being the birthplace of democracy, Western philosophy, Western literature, historiography, political science, major scientific and mathematical principles, theatre and the Olympic Games. From the eighth century BC, the Greeks were organised into various independent city-states, known as poleis (singular polis), which spanned the Mediterranean and the Black Sea. Philip II of Macedon united most of present-day Greece in the fourth century BC, with his son Alexander the Great rapidly conquering much of the ancient world, from the eastern Mediterranean to India. The subsequent Hellenistic period saw the height of Greek culture and influence in antiquity. Greece was

annexed by Rome in the second century BC, becoming an integral part of the Roman Empire and its continuation, the Byzantine Empire, which was culturally and linguistically predominantly Greek. The Greek Orthodox Church, which emerged in the first century AD, helped shape modern Greek identity and transmitted Greek traditions to the wider Orthodox world. After falling under Ottoman dominion in the mid-15th century, Greece emerged as a modern nation state in 1830 following a war of independence. The country's rich historical legacy is reflected in part by its 18 UNESCO World Heritage Sites.

Greece is a unitary parliamentary republic, and a developed country, with an advanced high-income economy, and a high quality of life, ranking simultaneously very high in the Human Development Index. Its economy is the largest in the Balkans, where it is an important regional investor. A founding member of the United Nations, Greece was the tenth member to join the European Communities (precursor to the European Union) and has been part of the Eurozone since 2001. It is also a member of numerous other international institutions, including the Council of Europe, the North Atlantic Treaty Organization (NATO), the Organisation for Economic Co-operation and Development (OECD), the World Trade Organization (WTO), the Organization for Security and Co-operation in Europe (OSCE), and the Organisation internationale de la Francophonie (OIF). Greece's unique cultural heritage, large tourism industry, prominent shipping sector and geostrategic importance classify it as a middle power.[c]”

6.3.2.3 Summarized Text:

“Greece (Greek: Ελλάδα, romanized: Elláda, [eˈlaða]), officially the Hellenic Republic,[b] is a country located in Southeast Europe. Its population is approximately 10.7 million as of 2018; Athens is its largest and capital city, followed by Thessaloniki. Situated on the southern tip of the Balkans, Greece is located at the crossroads of Europe, Asia, and Africa. Philip II of Macedon united most of present-day Greece in the fourth century BC, with his son Alexander the Great rapidly conquering much of the ancient world, from the eastern Mediterranean to India.”

6.3.3 Example 3

6.3.3.1 Parameters and Stats:

- Initial Word Number: 190
- Final Word Number: 102

- Reduction: 46%
- Model: distilbert-base-uncased
- No Sentences: 3
- Min Length: 25
- Max Length: 500
- Total time: 10.93s

6.3.3.2 Initial Text:

“Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data.[1] It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[3]

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning.[5][6] Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain.[7][8] In its application across business problems, machine learning is also referred to as predictive analytics.”

6.3.3.3 Summarized Text:

“Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. [1] It is seen as a part of artificial intelligence. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. [5][6] Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain. [7][8] In its application across business problems, machine learning is also referred to as predictive analytics.”

7

Technical details

7.1 Libraries

7.1.1 Python and Flask

Python is a general-purpose, high-level programming language that focuses on the code readability, for web development lines of code will be fewer than other languages. It is possible for Python because of large standard libraries which make the web development code simple and short. These libraries have pre-coded functions provided by Python community which can be easily downloaded and can be used as per the development needs. Initially Python was designed for web servers to deal with the incoming traffic on the server. Flask is a micro framework of Python which provides the basic functionality of web framework and allows more plug-ins to be added so the functionality and feature set can be extended to a new level. Flask is called as micro framework of Python because it makes the core functionality simple but extensible in terms of development. It can also be used to save time building web applications. Flask uses Jinja Template Engine and the Werkzeug WSGI Toolkit. Flask structure is categories into two parts “Static files & Template files”, template file have all the Jinja templates including Html pages, where as static file have all static codes needed for website such as CSS code, JavaScript code and Image files. [13]

7.1.1.1 Flask Web Framework

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask’s framework is more explicit than Django’s framework and is also easier to learn because it has less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread

management, etc. Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.

7.1.1.2 Flask-WTF

WTF stands for WT Forms which is intended to provide the interactive user interface for the user. The WTF is a built-in module of the flask which provides an alternative way of designing forms in the flask web applications. WTF is useful due to the following factors.

- The form elements are sent along with the request object from the client side to the server side. Server-Side script needs to recreate the form elements since there is no direct mapping between the client side form elements and the variables to be used at the server side.
- There is no way to render the HTML form data at real time.

The WT Forms is a flexible, form rendering, and validation library used to provide the user interface.

7.1.1.3 Jinja2

Jinja is a web template engine for the Python programming language. It was created by Armin Ronacher and is licensed under a BSD License. Jinja is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code. The Jinja template engine allows customization of tags, filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects. Jinja is Flask's default template engine.

7.1.2 Bootstrap

Bootstrap, originally named Twitter Blueprint, was developed by Mark Otto and Jacob Thornton at Twitter as a framework to encourage consistency across internal tools. Before Bootstrap, various libraries were used for interface development, which led to inconsistencies and a high maintenance burden. According to Twitter developer Mark Otto:

“A super small group of developers and I got together to design and build a new internal tool and saw an opportunity to do something more. Through that process, we saw ourselves build something much more substantial than another internal tool. Months later, we ended up with an early version of Bootstrap as a way to document and share common design patterns and assets within the company.” [10]

After a few months of development by a small group, many developers at Twitter began to contribute to the project as a part of Hack Week, a hackathon-style week for the Twitter

development team. It was renamed from Twitter Blueprint to Bootstrap, and released as an open source project on August 19, 2011.It has continued to be maintained by Mark Otto, Jacob Thornton, and a small group of core developers, as well as a large community of contributors [11].

7.1.3 NLTK

The Natural Language Toolkit (NLTK) was developed in conjunction with a computational linguistics course at the University of Pennsylvania in 2001 (Loper and Bird, 2002). It was designed with three pedagogical applications in mind: assignments, demonstrations, and projects.

- Assignments. NLTK supports assignments of varying difficulty and scope. In the simplest assignments, students experiment with existing components to perform a wide variety of NLP tasks. As students become more familiar with the toolkit, they can be asked to modify existing components, or to create complete systems out of existing components.
- Demonstrations. NLTK's interactive graphical demonstrations have proven to be very useful for students learning NLP concepts. The demonstrations give a step-by-step execution of important algorithms, displaying the current state of key data structures.
- Projects. NLTK provides students with a flexible framework for advanced projects. Typical projects might involve implementing a new algorithm, developing a new component, or implementing a new task. [12]

7.2 Platforms and Coding Tools

This describes the characteristics of this implementation, such as the development and execution platform, programming tools, hardware application requirements, etc. Be careful to provide all the details, including necessary software and settings.

7.2.1 PycharmCE

For the development of the application, we used PyCharm. PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems.

PyCharm is available in three editions: Professional, Community, and Edu. The Community and Edu editions are open-source projects and they are free, but they have fewer features. PyCharm Edu provides courses and helps you learn programming with Python. The Professional edition is commercial, and provides an outstanding set of tools and features.

7.2.2 Python pip

For the management of our Python packages, we used pip. Pip it's a tool that allows you to install and manage additional libraries and dependencies that are not distributed as part of the standard library. Package management is so important that pip has been included with the Python installer since versions 3.4 for Python 3 and 2.7.9 for Python 2, and it's used by many Python projects, which makes it an essential tool for the development.

7.2.3 Python venv

In order to isolate the development environment from the original python installation we leveraged the venv module that provides support for creating lightweight "virtual environments" with their own site directories, optionally isolated from system site directories. Each virtual environment has its own Python binary (which matches the version of the binary that was used to create this environment) and can have its own independent set of installed Python packages in its site directories.

8

Epilogue

8.1 Summary and conclusions

The web application we built is consists of a simple UI that could give the opportunity for common users to summarize their texts easily. Below we will discuss some potential improvements that can help the application to scale and also improve the user's experience and enhance or extend our use cases and consumption model.

8.2 Future extensions

In this section we will describe some further enhancements of our application for the future and how we can make it to be able to run at scale, more efficiently and possible integration with different models like BERT.

8.2.1 Application performance cache

One of the improvements that could be done is related to performance of the application. In order to avoid running high computational tasks over and over again we could hash the texts along with the parameters used and store the document in a database for permanent storage where the data will stay at rest [7]. This will help our application to scale as more users will request to summarize more texts and the demand will increase, we could leverage the database and then move the data into cache in-memory. There are various technologies for this implementation like Redis or Memcached and multiple scenarios to configure.

Benefits of performance cache could be:

- Performance: users will access request really fast. In our case the performance gain will be huge as an average request takes about 10-12 secs and in case another user requests exactly the same summary in the future this will be reduced to less than a second.

- Cost Reduction, our model consumes high level of GPU or CPU for every request apart of the time it takes, we could potentially reduce the overall cost if we store the data into disk and instead of re-calculation, we can just send a query to a DB. A common Architecture design of web cache could be found in the next diagram (see Figure 8-1).

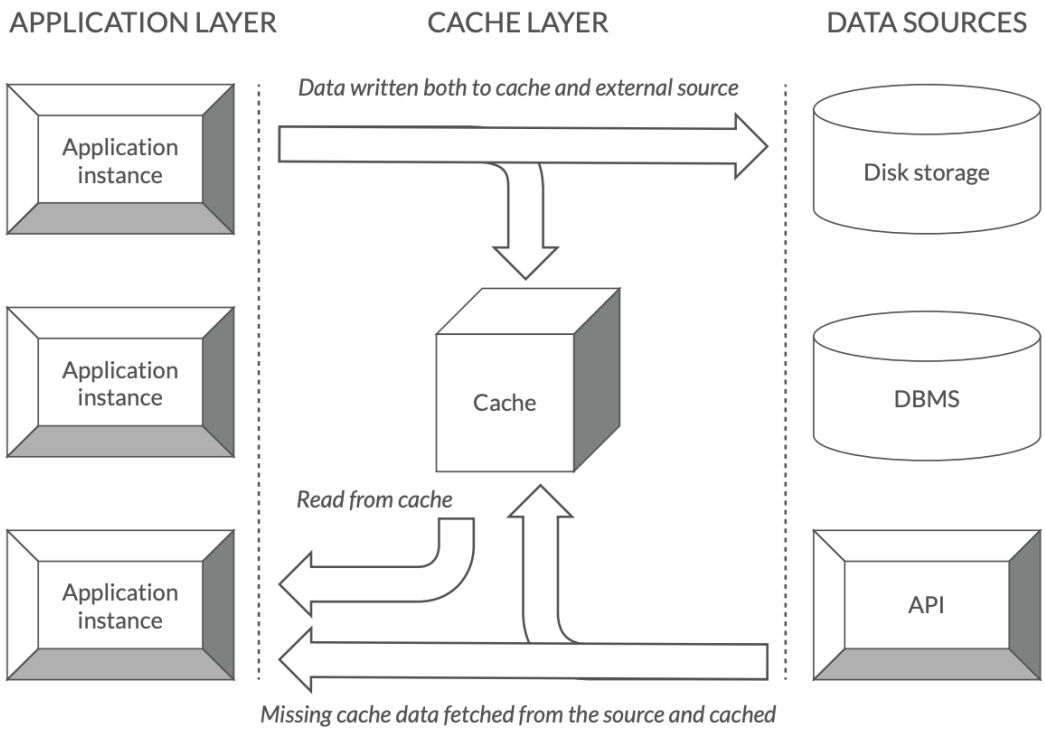

Figure 8-1 Web Cache Architecture [7]

8.2.2 Web Application Programming Interface (API)

Another potential improvement is the implementation of Application Programming Interface (API). This will help our users to leverage the existing functionality of our application and interact with this via a web API. In order to make sure we can support multiple application at high scale we can implement a rate limiter or throttling mechanism to distribute requests evenly across different applications and users.

8.2.3 Integration with a Queue

In order to implement prioritization of background tasks that our text summarization application runs, we could integrate our backend with a queuing system like RabbitMQ [7]. The overall idea is to add an extra functionality where tasks could run asynchronously and also have different priorities. With this kind of integration, we could easily manage huge number of requests that are coming through the front-end or the API of our implementation and distribute them across different compute instances in case we have load balancing. A common

architecture of a Web application that uses an API and is built based on microservices could be found in the “Figure 8-2 Microservice web application with RESTful API”.

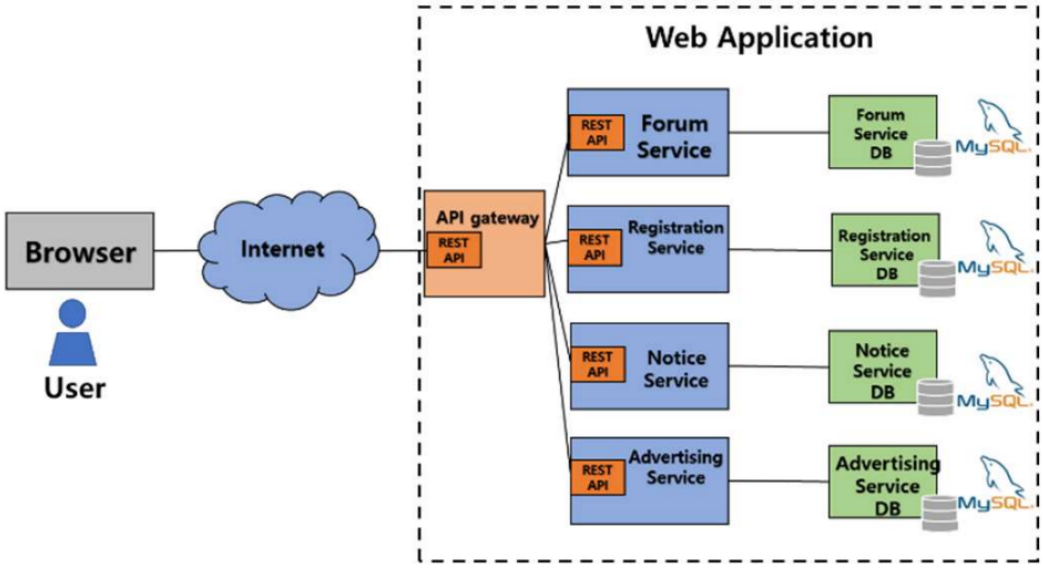

Figure 8-2 Microservice web application with RESTful API [8]

8.2.4 Load Balancing

Load balancing (LB) is the method of distributing the user requests throughout different instances of our application. The main reason we need load balancing is to accept huge amount of request that could be impossible to be handled by a single instance. The idea is to have multiple instances of different layers of the application like front-end servers, back-end servers and distribute the traffic to give our users the best experience we can. A high-level idea of a 3-tier web application that uses load balancing could be found in “Figure 8-3 Typical 3-tier web service architecture” [8]

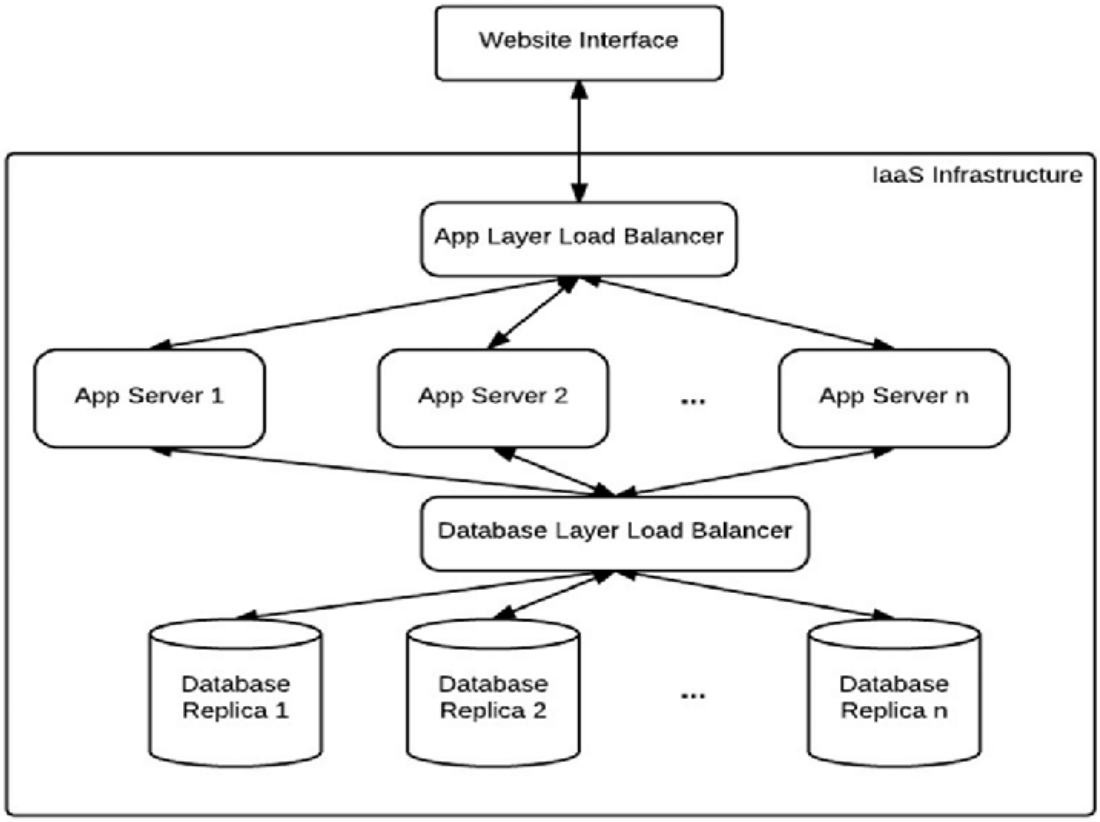

Figure 8-3 Typical 3-tier web service architecture [9]

9

Bibliography

[1] M. C. Kenton, L. Kristina, and J. Devlin, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” no. Mlm, 1953.

[2] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Computational linguistics 28, 4 (2002), 399–408.

[3] I. International, C. Conference, E. Emerging, T. Trends, E. Engineering, and T. Technology, “A survey of Extractive and Abstractive Automatic Text Summarization Techniques,” pp. 109–110, 2013.

[4] M. Allahyari, E. D. Trippe, and J. B. Gutierrez, “Text Summarization Techniques: A Brief Survey,” no. 1.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT , a distilled version of BERT : smaller , faster , cheaper and lighter,” pp. 2–6, 2019.

[5] S. Brown, “Machine Learning - Explained,” 2021. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained. [Accessed: 25-Jul-2021].

[6] Sentiment Analysis of COVID-19 Vaccination from Survey Responses in Bangladesh. - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Architecture-of-Bidirectional-Encoder-Representations-from-Transformers-BERT-31_fig3_351393481 [accessed 10 Sep, 2021]

[7] E. Summary, “whitepaper 15 Reasons to use Redis as an Application cache”

[8] M. C. Kenton, L. Kristina, and J. Devlin, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” no. Mlm, 1953.

[9] The Impact of Database Layer on Auto-Scaling Decisions in a 3-Tier Web Services Cloud Resource Provisioning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Typical-3-tier-web-service-architecture_fig1_319643261 [accessed 01 Sep, 2021]

[10] Otto, Mark (January 17, 2012). "Bootstrap in A List Apart No. 342". Mark Otto's blog. from: https://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/ [Accessed 01 Sep, 2021]

[11] “Bootstrap (front-end framework)” [Online] Available: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Accessed: 30-Aug-2021].

[12] S. Bird and E. Loper, “NLTK : The Natural Language Toolkit.”

[13] F. A. Aslam, H. N. Mohammed, J. Musab, and M. Munir, “International Journal of Advanced Research in Computer Science Available Online at www.ijarcs.info Efficient Way Of Web Development Using Python And Flask,” vol. 6, no. 2, pp. 54–57, 2015.

[14] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A MULTI -TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND -,” pp. 1–20, 2019.

[15] D. Miller, “Bert Extractive Summarizer.” [Online]. Available: https://github.com/dmmiller612/bert-extractive-summarizer. [Accessed: 10-Aug-2021].

[16] El Naqa I., Murphy M.J. (2015) What Is Machine Learning?. In: El Naqa I., Li R., Murphy M. (eds) Machine Learning in Radiation Oncology. Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3_1

[17] Jones K.S. (1999) What is the Role of NLP in Text Retrieval?. In: Strzalkowski T. (eds) Natural Language Information Retrieval. Text, Speech and Language Technology, vol 7. Springer, Dordrecht. https://doi.org/10.1007/978-94-017-2388-6_1