Cairo University
Faculty of Computers and Information
Department of Computer Sciences

Text Classification

Supervised by

Dr. Hesham Hassan

TA. Dalia Maher

Implemented by

ID	Name	E-mail	Group
20170022	Ahmed Sayed Mansour	ahmed.mans20719@gmail.com	CS-IS-1
20170021	Ahmed Sayed Ibrahim	ahmed111522@gmail.com	CS-DS-1
20170136	Atef Magdy Mitwally	atefmagdy12@gmail.com	CS-IS-1
20170053	Ashraf Samir Ali	ashrafsamer423@gmail.com	CS-DS-1
20170002	Ibrahim Ramadan Abdou	ibrahemramadan130@gmail.com	CS-IS-1

Graduation Project
Academic Year 2020-2021
Mid-year Short Documentation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
NLP	Natural language Processing
SVM	Support Vector Machine
NB	Naive Bayes
TF-IDF	Term frequency-inverse document frequency
BOW	Bag of Words
SVC	Support Vector Classifier
RF	Random Forest
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network

Abstract

With the explosion of information fueled by the growth of the numbers of text categories. It is no longer feasible for a human observer to understand all the data coming in or even classify it into categories. With this growth of information and simultaneous growth of available computing power automatic classification of data, particularly textual data, gains increasingly high importance. This project provides a review of the text classification process, phases of that process and methods being used at each phase.

Examples from research papers classification are provided throughout the text. Principles of operation of four main text classification algorithms “Naïve Bayesian”, “Neural networks”, “Support Vector Machines” and “Random forest”. This project will look through the state of the art in all these phases, take note of methods and algorithms used and of different ways that researchers are trying to reduce computational complexity and improve the precision of text classification process as well as how the text classification is used in practice.

The paper is written in a way to avoid extensive use of mathematical formulae in order to be more suited for readers with little or no background in theoretical mathematics.

1. Introduction

1.1. What is classification?

Classification is a supervised machine learning technique, Classification is the process of categorizing a given set of data into classes, and it can be performed on both structured and unstructured data. The process starts with predicting the class of given data points.

The classes are often referred to as target, label or categories.

1.2. What is text classification?

Text classification is the process of categorizing a text into organized groups. By using Natural Language Processing (NLP), text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content. As shown in figure 1.

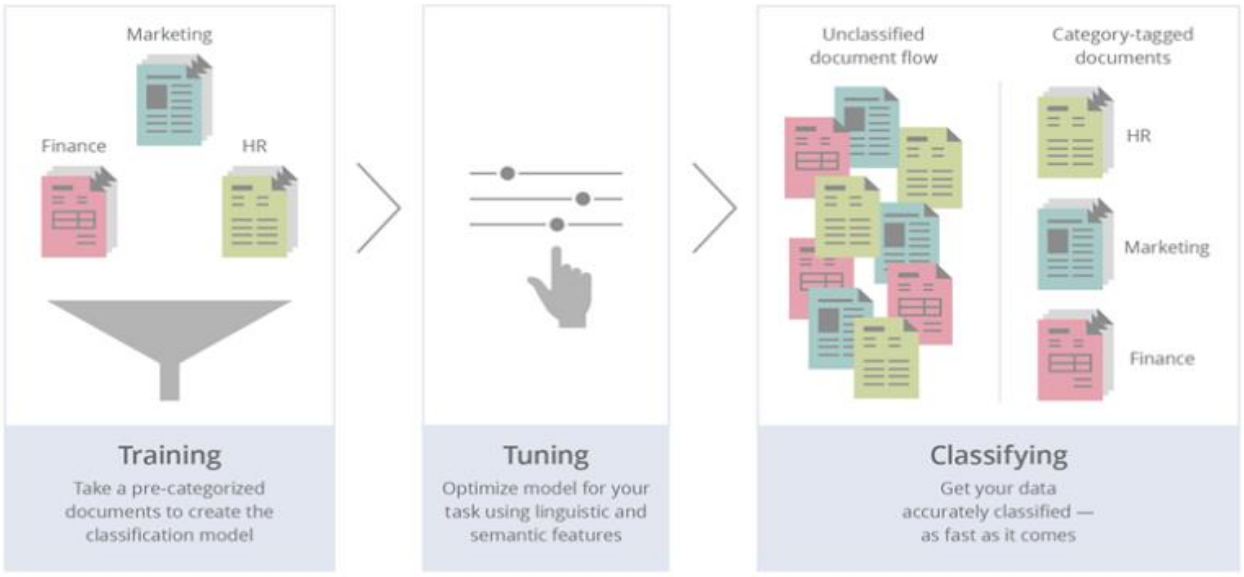

Figure 1 - Text Classification Processing

1.3. Variants of text classification

- **Binary categorization:** only two categories
  - Retrieval: {relevant-doc, non-relevant-doc}
  - Span filtering: {spam, not-spam}
  - Opinion: {positive, negative}
- **K-category categorization:** more than two categories
  - Topic categorization: {sports, science, travel, business}
  - Email routing: {folder1, folder2, folder3 …}
- **Hierarchical categorization:** categories from hierarchy.
- **Joint categorization:** multiple related categorization tasks done in joint matter.

1.4. Why is text classification important?

It's estimated that around 80% of all information is unstructured, with text being one of the most common types of unstructured data. Because of the messy nature of text, analyzing, understanding, organizing, and sorting through textual data is hard and time-consuming, so most companies fail to use it to its full potential. This is where text classification with machine learning comes in. Using text classifiers, companies can automatically structure all manner of relevant text, from emails, legal documents, social media, Chabot's, surveys, and more in a fast and cost-effective way. This allows companies to save time analyzing text data, automate business processes, and make data-driven business decisions. [1]

1.5. The uses of text classification

- **Language detection**: the procedure of detecting the language of a given text (e.g., know if an incoming support ticket is written in English or Spanish for automatically routing tickets to the appropriate team).
- **Topic Labeling classification**: the task of identifying the theme or topic of a piece of text (e.g., know if a product review is about Ease of Use, Customer Support, or Pricing when analyzing customer feedback).

- **Sentiment Analysis**: the process of understanding if a given text is talking positively or negatively about a given subject (e.g., for brand monitoring purposes).
- **Intent classification**: is another great use case for text classification that analyzes text to understand the reason behind feedback.[2][3]

1.6. Gantt chart of project time plan

As shown in figure 2, we see our done work from Sept 2020 until April 2021 starting with finding the idea of the project and ending with designing our machine learning models.

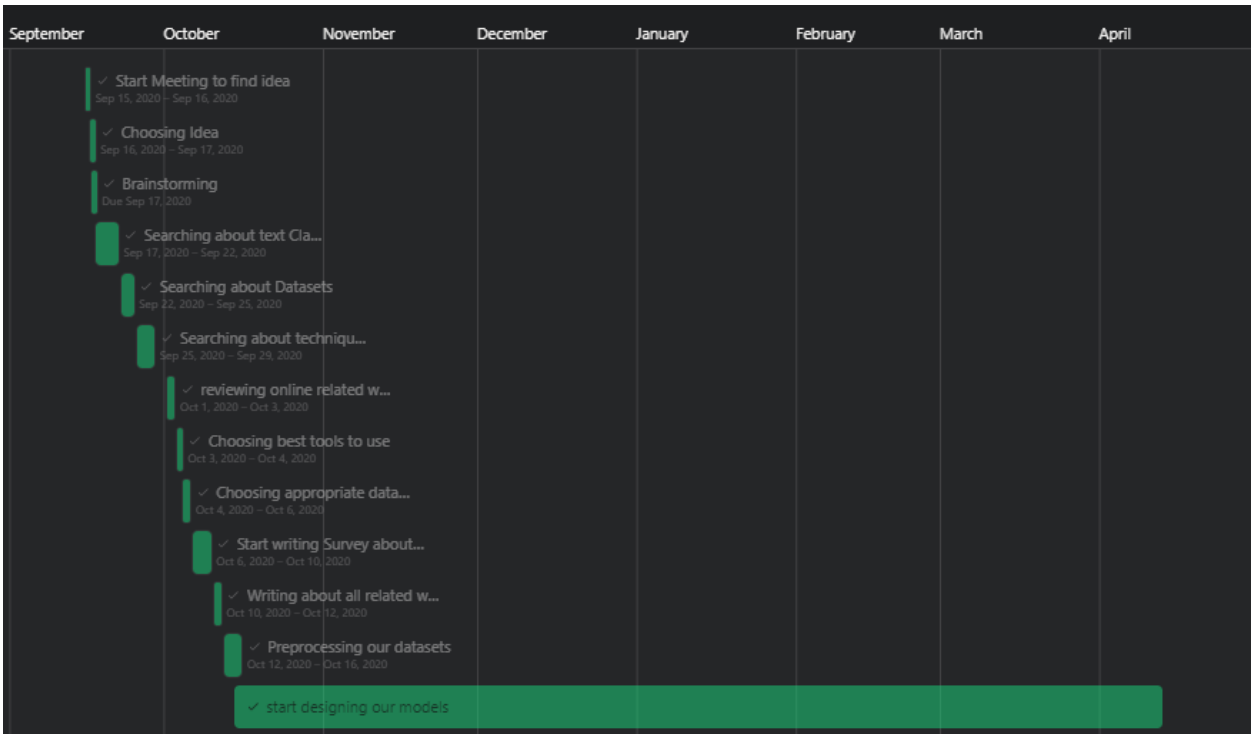

Figure 2 - Gantt chart I

As shown in figure 3, we see our done work from Oct 2020 until July 2021 starting with training ML models and ending with publishing our research.

Figure 3 - Gantt chart II

1.7. Objectives

Our objectives are to try different techniques in text classification to know which one is better and has the best efficiency on research papers datasets. We divided our objectives to these points:

- Building single-label text classification models.
- Building multi-label text classification models.
- Trying different techniques for classification models and differentiate between the different techniques based on (Accuracy, Performance, Memory handling).
- Building web application using react libraries as a front-end and Flask & Node JS as a back-end, to let people use our trained models.
- Posting our research on the web app to be public for developers and persons who is interested in text classification techniques.

1.8. Report Organization

Chapter 2: Background (more details about text classification, algorithms and techniques)

Chapter 3: Related work (analysis about the current text classification projects)

Chapter 4: Proposed text classification (project workflow, architecture and algorithms)

Chapter 5: System Analysis (methodology, requirements, use-cases, diagrams)

Chapter 6: System design (System Component Diagram, System Class diagram, Sequence Diagram, Project ERD, System GUI Design)

Chapter 7: Implementation and testing (Models Results, Back end (APIs - Server), Test Cases)

- Conclusion
- References

2. Background

In this chapter, we first review the methodologies for text classification. Concretely, we illustrate the **Preprocessing-data**, **NLP** of text classification, **algorithms** and **technologies**. Preprocessing-data includes text representation. For text representation, we review how to extract features from text, how to clean text and in NLP we review most known techniques for Text-NLP**.** Lastly, we review some algorithms and technologies used for classification implementation.

2.1. Preprocessing Data

In natural language processing, text preprocessing is the practice of cleaning and preparing text data.

- **Noise Removal:** noise removal is a text preprocessing task devoted to stripping text of formatting.
- **Tokenization:** tokenization is the text preprocessing task of breaking up text into smaller components of text (known as tokens).
- **Text Normalization:** normalization encompasses many text preprocessing tasks including stemming, lemmatization, upper or lowercasing, and stop-words removal.
- **Stemming:** stemming is the text preprocessing normalization task concerned with bluntly removing word affixes (prefixes and suffixes).
- **Lemmatization:** lemmatization is the text preprocessing normalization task concerned with bringing words down to their root forms.
- **Stop-word Removal:** stop-word removal is the process of removing words from a string that don't provide any information about the tone of a statement.[4]

2.2. Nature Language Processing

Natural Language Processing (**NLP**) is a field of Artificial Intelligence (AI) that makes human language intelligible to machines. NLP combines the power of linguistics and computer science to study the rules and structure of language, and create intelligent systems (run on machine learning and NLP algorithms) capable of understanding, analyzing, and extracting meaning from text and speech. There are many approaches to NLP text classification, which fall into three types of systems:

- **Rule-based systems:** In the rule-based approach, texts are separated into an organized group using a set of handicraft linguistic rules. Those handicraft linguistic rules contain users to define a list of words that are characterized by groups. For example, words like Donald Trump and Boris Johnson would be categorized into politics. People like LeBron James and Ronaldo would be categorized into sports**.**
- **Machine learning-based systems:** Machine-based classifier learns to make a classification based on past observation from the data sets. User data is pre labeled as train and test data. It collects the classification strategy from the previous inputs and learns continuously. Machine-based classifier usage a **bag of a word** for feature extension. It's preferred to use **tf–idf algorithm** to normalize the number of repeated words. As shown in figure 4.

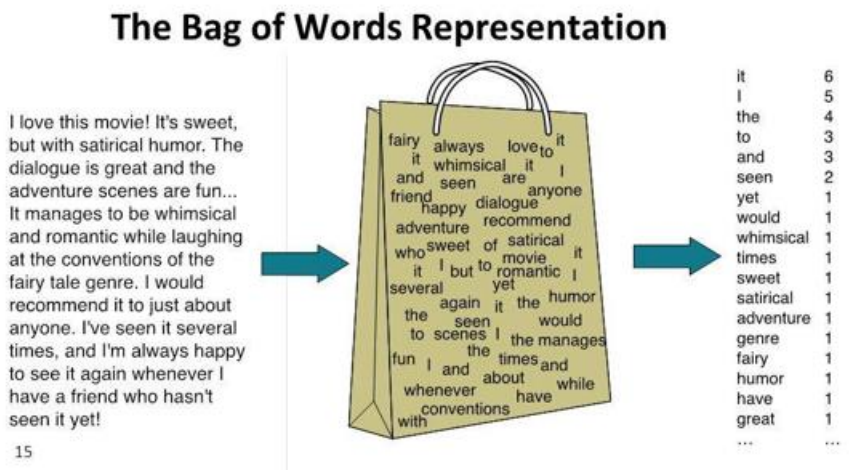

Figure 4 - Feature Selection (Bag of Words Representation)

- **Hybrid systems:** Hybrid approach usage combines a rule-based and machine Based approach. Hybrid based approach usage of the rule-based system to create a tag and use machine learning to train the system and create a rule. Then the machine-based rule list is compared with the rule-based rule list. If something does not match on the tags, humans improve the list manually. It is the best method to implement text classification.[5][6]

2.3. Algorithms

We separated our problem into two categories:

- **Single-label text classification:** Is the task of classifying the elements of a set into two groups on the basis of a classification rule. Some of the most popular machine learning algorithms for creating text classification models include the **Naive Bayes** family of algorithms, **support vector machines (SVM),** and **deep learning**.

- **Multi-label text classification:** is a generalization of multiclass classification, which is the single-label problem of categorizing instances into precisely one of more than two classes; in the multi-label problem there is no constraint on how many of the classes the instance can be assigned to. Some of the most popular techniques **One Vs Rest** to use with any of the mentioned algorithms above**.** [7]

3. Related work

For our research we will discuss some research that talks about the idea and shows the differences.

3.1. Resources

Resource 1: [8]

Problem: Binary classification on **The 20 Newsgroups data set.**

Solution: Using scikit-learn and NLTK to load the data and for the NLP process using sklearn.feature_extraction library to apply Count Vectorization and TF-IDF Transformer to the data. Building the model using Naive Bayes algorithm with accuracy of 69% and using SVM with accuracy of 68%.

Advantages:

- Using different algorithms from the sklearn library.
- Explaining all steps and reporting information on each experiment.

Disadvantages:

- The data set is small and only covers a little number of classes.
- Not removing all unimportant data like headers and footers.
- Not using the new trends of text classification like Neural Networks algorithms.

Resource 2: [9]

Problem: Discussing best pre trained models for text Classification

Solution: Using pre-trained models like (XLNet, ERNIE, T5…) on different datasets.

Advantages:

- The datasets meet industry-accepted standards.
- The pre-trained models have already been vetted on the quality aspect.
- Reporting a summary on each model using different datasets.

Disadvantages: Using the most new trends of ML and deep learning, which needs a lot of studying to understand.

3.2. The main differences between related works and our Project

- All related work working on small datasets while we using the Arxiv data (3 GB).
- Our training on many numbers of labels to cover a wider area of subjects.
- Letting users to try our models on our website.
- Our training for Multilayer classification for better performance not only single labels on the related work.

4. Proposed text classification

The steps for our model, and how to prepare the data for the models.

4.1. Workflow for trained models

4.1.1. Preprocessing data

First phase is cleaning data from any unimportant words and characters in many steps as shown in figure 5.

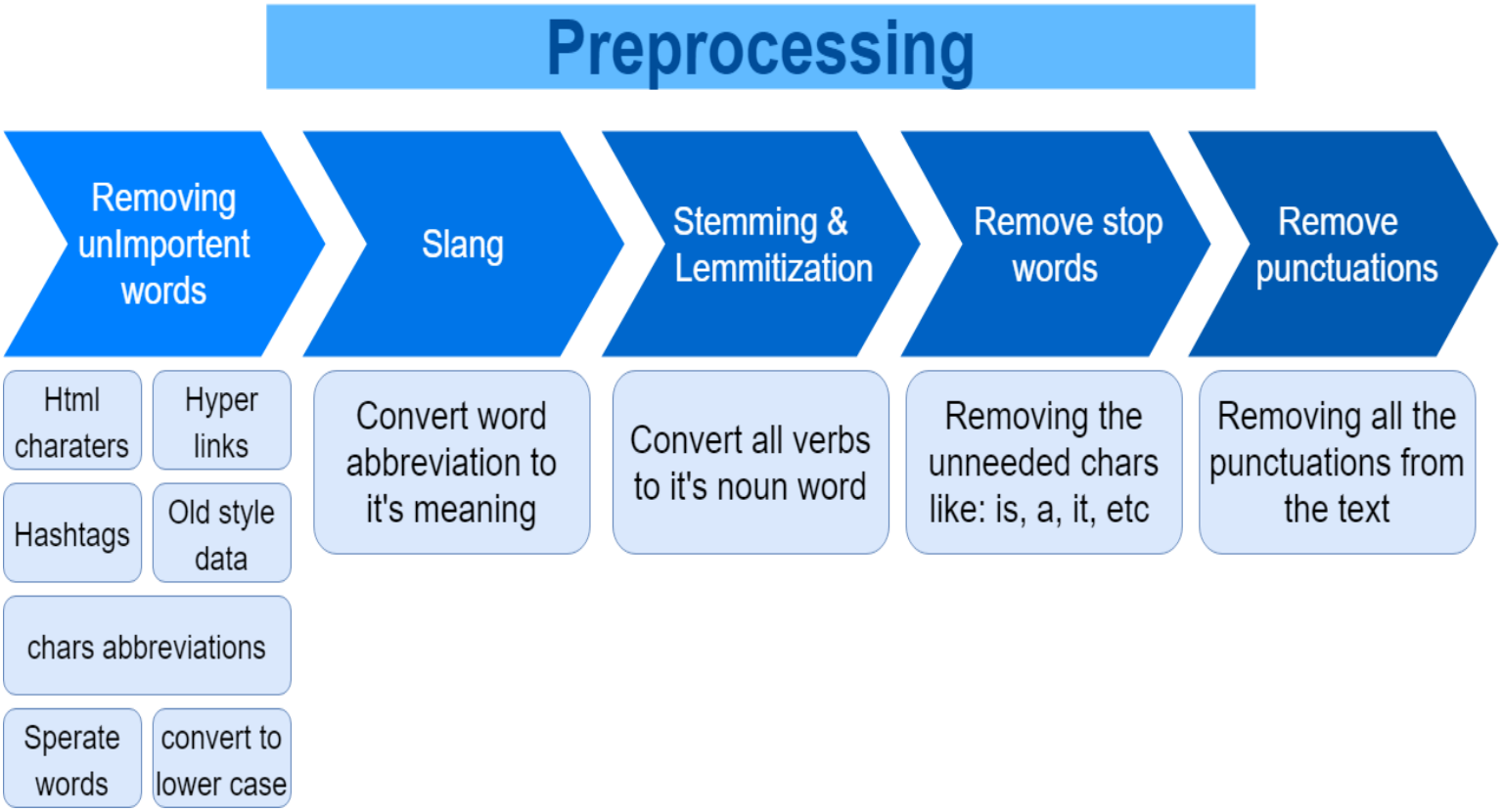

Figure 5 - Preprocessing data

4.1.2. Natural Language Processing (NLP)

Second phase is to convert textual data into numerical data to be used later.

4.1.2.1. Text Vectorization

This is the first step to proceed NLP by selecting all words from files by

“Bag of words technique” and give each file a vector of numbers as shown in Fig 6.

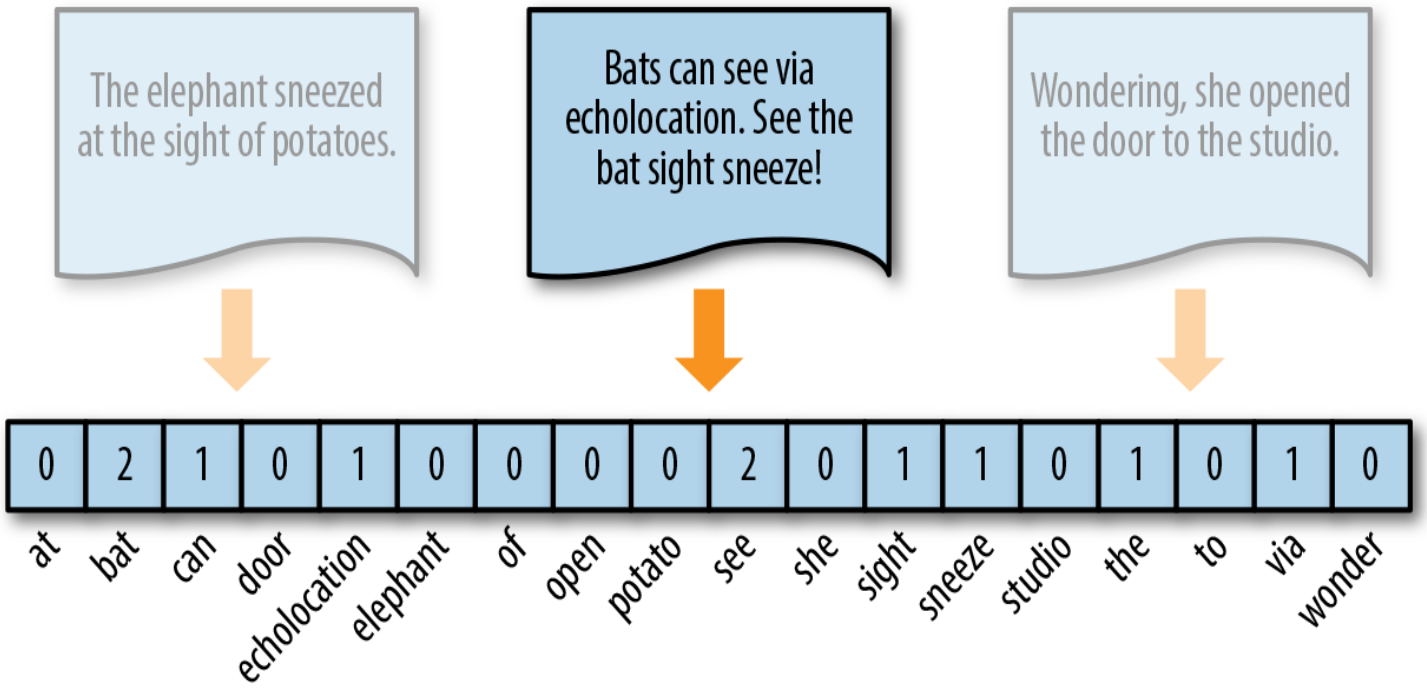

Figure 6 - Text vectorization

4.1.2.2. TF-IDF Normalization

This is the second step of NLP by normalizing previously mentioned vectors to improve the performance of the learning phase as shown in table 1.

Table 1 - TF-IDF Normalization

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

4.1.3. Training and Prediction

Last phase is to:

- Train our model by using extracted vectors from last 2 phases with their label to generate classifier model.
- Use generated classifier model to predict new untrained data.

As shown in figure 7.

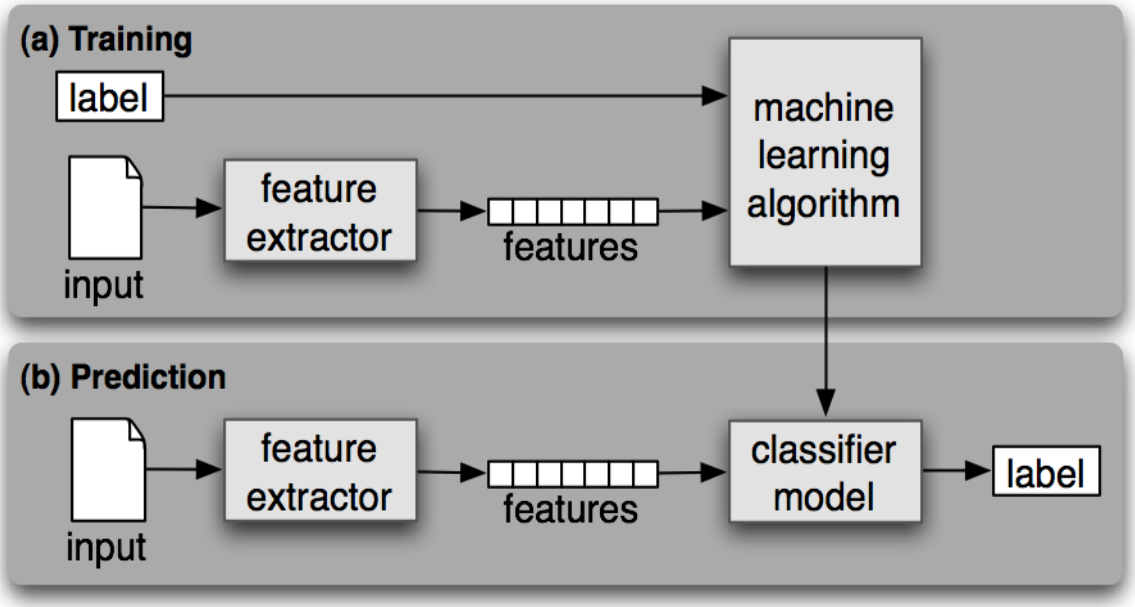

Figure 7 - Text classification processing

4.2. Workflow for Pre-trained models

4.2.1. Loading Pre-trained models

4.2.1.1. Architecture

The architecture provides the working parameters—such as the number, size, and type of layers in a neural network.

4.2.1.2. Tokenizer

The Tokenizer, which tokenized the pre-trained vocab, and loaded to tokenize the new dataset, as shown in Figure 8.

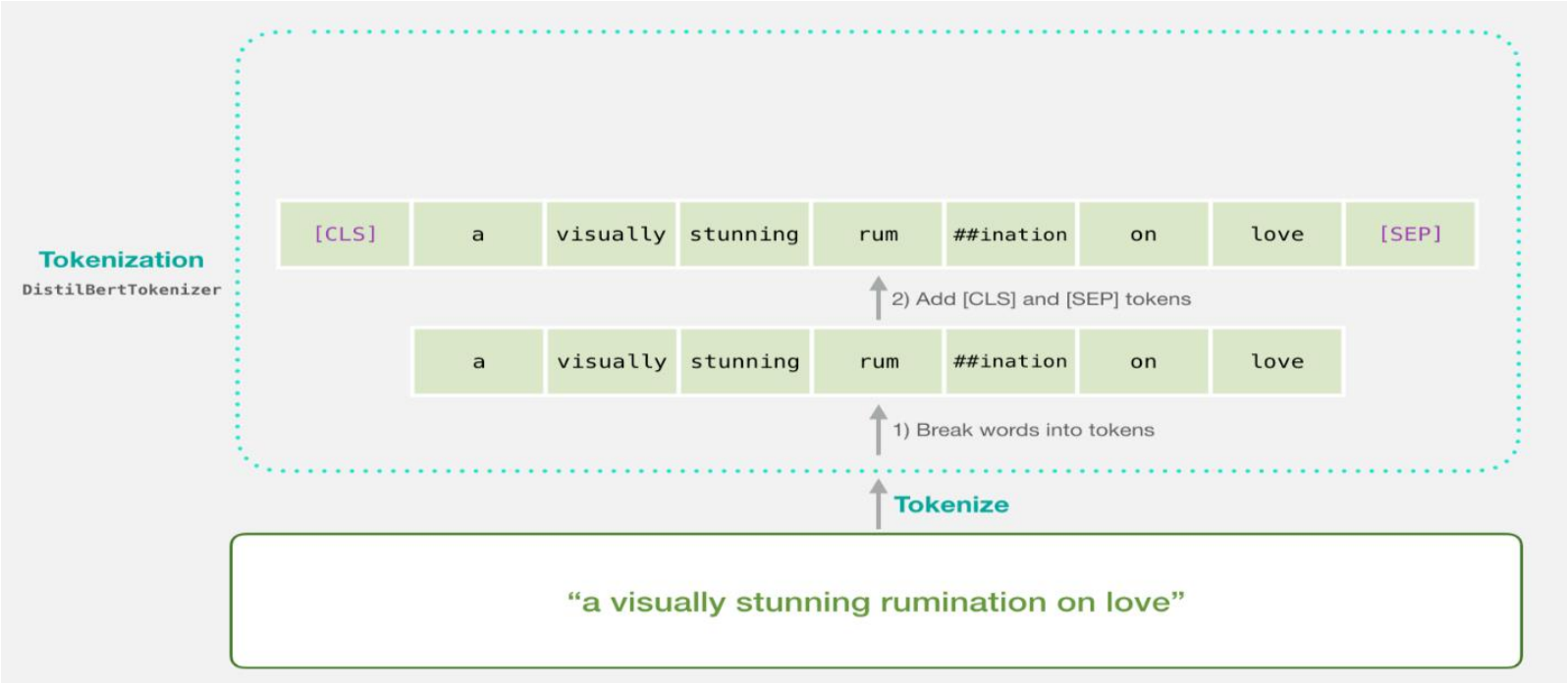

Figure 8 - Pre-Trained Tokenization

4.2.1.3. Vocab

The existing vocab from a pre-trained model to add the new dataset vocabs to it as shown in figure 9.

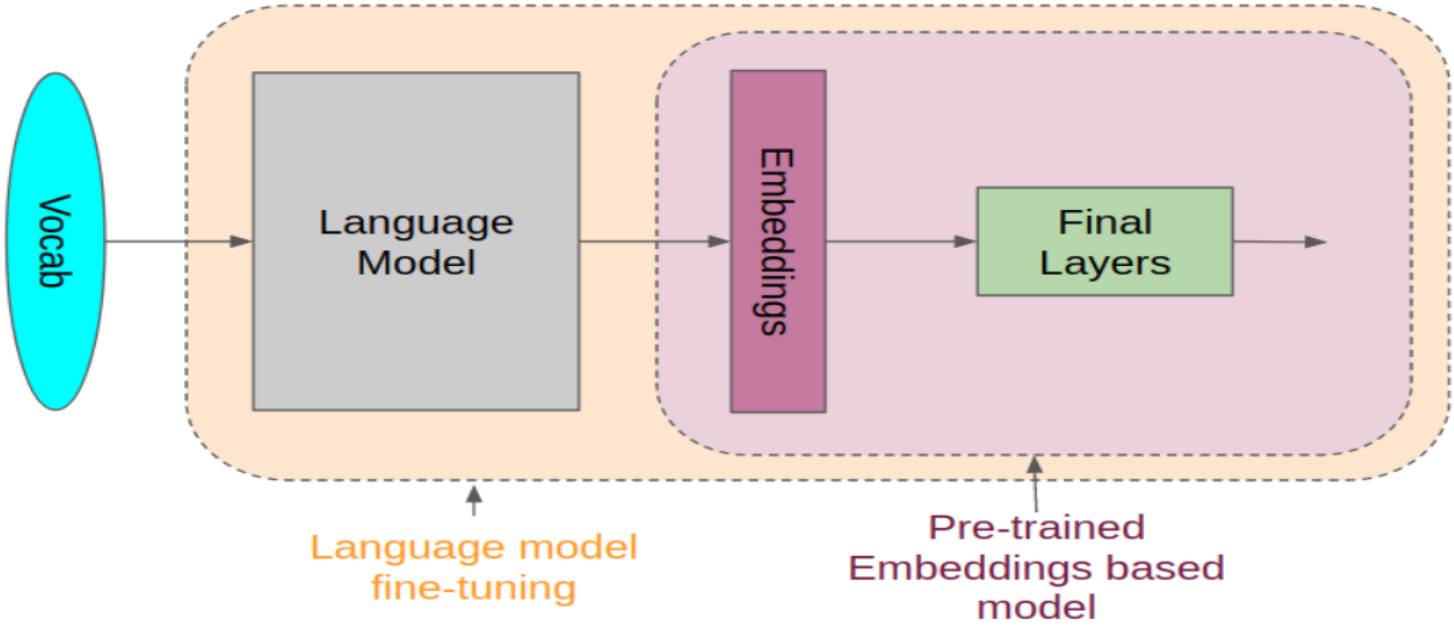

Figure 9 - Pre-Trained Vocab

4.2.2. Preprocessing dataset

Use the loaded tokenizer to tokenize the new dataset into vocabs to use them in fine-tuning the pre-trained model

4.2.3. Fine-tune model

As shown in figure 10 and 11 in the steps:

i. Define the new Number of output labels
ii. Define the new arguments like (Number of epochs, learning rate, batch size, etc.)
iii. Compile the new model with the modified arguments
iv. Fit the dataset
v. Evaluate.

Figure 11 - Pre-training cycle

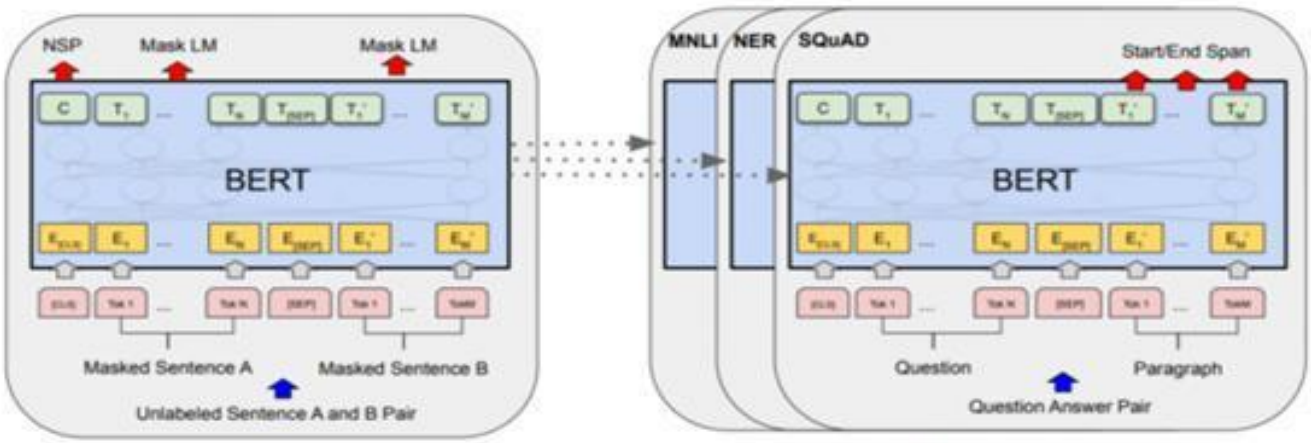

Figure 10 - Fine-tune model

5. System Analysis

5.1. Methodology

- **Agile Methodology**

Why **Agile** (**Extreme Programming):**

- To empower out team to manage our project easily.
- Flexible in accepting changes for our future work like adding new algorithms and more features.
- Parts of the often system can be deployed sooner.
- Many tough problems can be addressed early in the project.

- **Technologies**

  - Machine Learning (Python – Sklearn – numpy – TensorFlow – Keras - Pickle).
  - Natural Language Processing (Python – NLTK).
  - Front-End (Web standard W3C – REACT).
  - Back-End (Python – Node JS – Flask).
  - Testing (Jest - Pandas).

5.2. Domain

- Artificial Intelligence
- Web software
- Scientific applications

5.3. Stakeholders

- Users (people who wanted to use this app to classify their document).
- Interested third parties (people who have interest in the Text classification field and want to use our app on their projects/researches).

5.4. System Architecture

As shown in figure 12 our system architecture is all about using our service (ML model) to show the layers of our application.

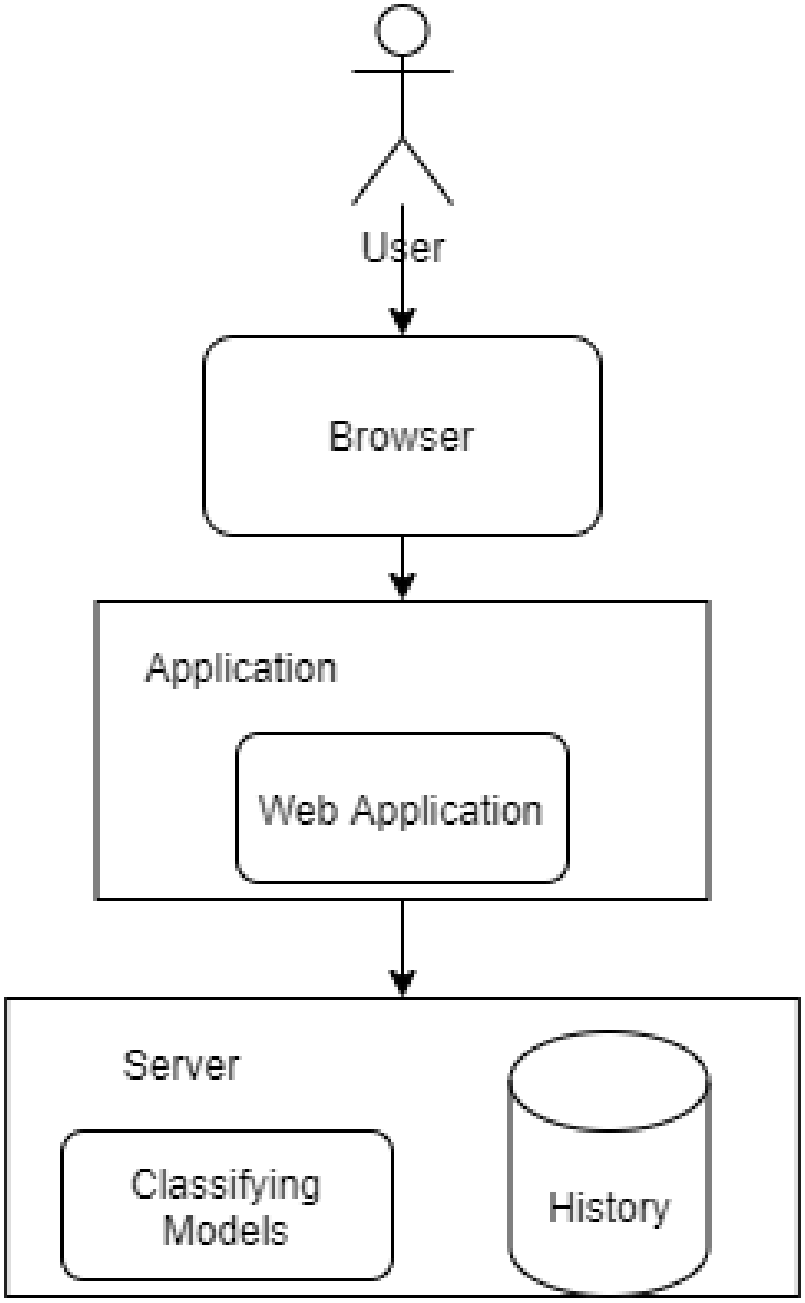

Figure 12 - System Architecture diagram

5.5. Functional and non-functional requirements

5.3.1. Functional Requirement

- Users can classify their document by writing or pasting it directly.
- System takes the document and parses its text to sentences, paragraphs and words.
- System uses a classifier or multi-classifiers to classify the document.
- System output class or multi-classes of the document.
- Reporting our research on Text-Classification on our website.
- User can browse the history of his classified documents

5.3.2. Non-Functional Requirements

Performance	Calculation time and response time should be as little as possible, because one of the software's features is timesaving. The classifier takes a maximum response time of 5 seconds to predict the input data.
Usability	The system should be easy to use. The user should reach the summarized text with one or two button press if possible. The user does not need time to learn and train to use out web application. The system also should be user friendly.
Reliability	The classifier is developed with machine learning, feature engineering and deep learning techniques, So, in this step there is no certain reliable percentage that is measurable. The web application should handle and work under a lot of requests and large data input without failure.

Scalability	The system is designed to be scalable to add and remove algorithms and features.

5.4. Use-Case Diagram

As shown in figure 13, we have multiple use-cases for different features of our app.

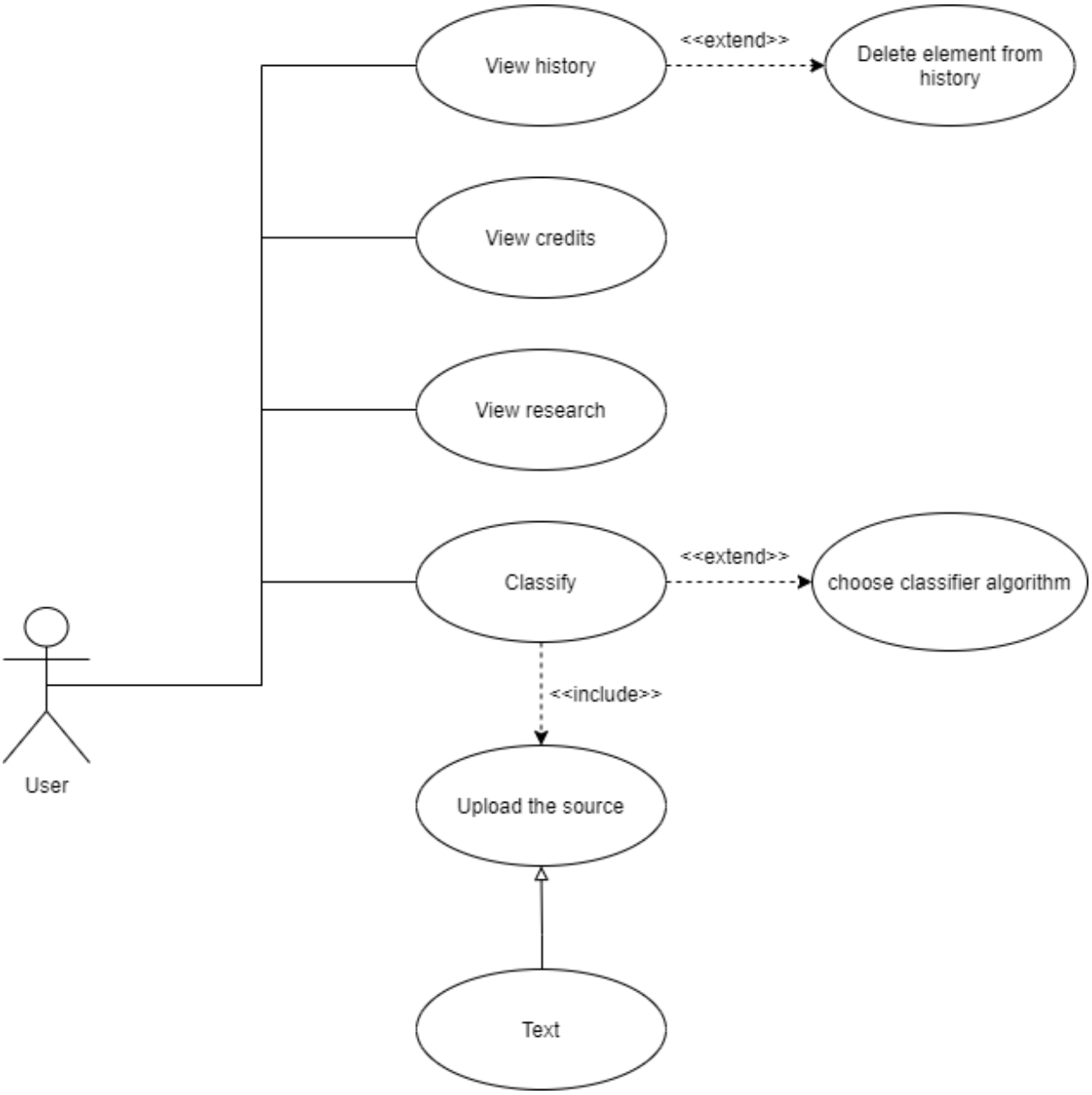

Figure 13 - Use-Case Diagram

5.5. Use-Case Description

Table 2 - Upload and classify a Document Use-Case

Use Case ID:	1	
Use Case Name:	Upload and classify a Document	
Actors:	User	
Pre-conditions:	Document Should be text based *******	
Post-conditions:	The document is ready to be classified	
	User Action	**System Action**
	1- User uploads a documents or type a text.	
	2- User request to classify the document or the input text.	
		3- System validate user input and Preprocess user input.
		4- System classifies the document or the user input.
	User Action	**System Action**

	1- User uploads a non-text input.	
		2- Response to user invalid input format.

Table 3 - Add history element Use-Case

Use Case ID:	2	
Use Case Name:	Add history element	
Actors:	User	
Pre-conditions:	Classification is done	
Post-conditions:	History element is added	
	User Action	**System Action**
	1- User choose if classification was right or wrong	
		2- system adds history element to database

Exceptions:	**User Action**	**System Action**
	1- User doesn't choose was classification right or wrong	
		2- system won't add this classification process into the database

Table 4 - View history Use-Case

Use Case ID:	3	
Use Case Name:	View history	
Actors:	User	
Pre-conditions:		
Post-conditions:	Getting list of history elements	
	User Action	**System Action**
	1- User click in history tab	
		2- system return a list of history elements of this user

Exceptions:	**User Action**	**System Action**
	1- User doesn't have any history elements	
		2- System will return a message that he doesn't have history yet.

Table 5 - Delete history element Use-Case

Use Case ID:	4	
Use Case Name:	Delete history element	
Actors:	User	
Pre-conditions:	User is in history tab	
Post-conditions:	History element deleted	
	User Action	**System Action**
	1- User click on delete history element	

		2- system delete this history element

Table 6 - View Credits Use-Case

Use Case ID:	5	
Use Case Name:	View credits	
Actors:	User	
Pre-conditions:		
Post-conditions:	Getting list of persons who have done the project	
	User Action	**System Action**
	1- User click on credits tab	
		2- system returns list of persons who have done the project

Use Case ID:	6

Use Case Name:	View research	
Actors:	User	
Pre-conditions:		
Post-conditions:	Getting our research paper so he could read it	
	User Action	**System Action**
	1- User click on “Our research” tab	
		2- system returns our research paper

6. System Design

6.1. System Component Diagram

As shown in Figure 14 our component diagram is separated into two components (History Component and Classifier Component) both are inherited from the System Component.

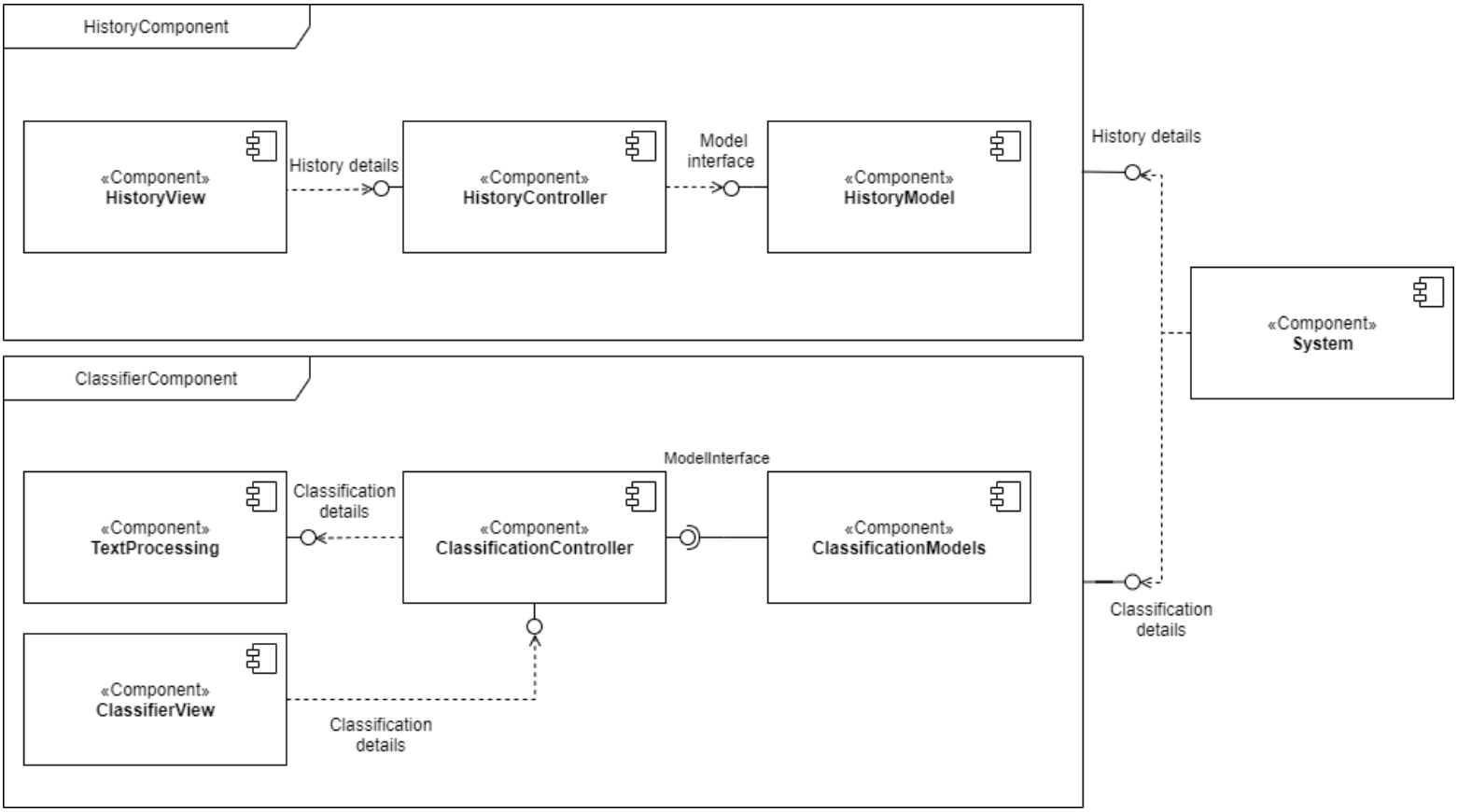

Figure 14 - Component Diagram

6.2. System Class diagram

As shown in Figure 15, in this class diagram we used the strategy design pattern to generalize the process of adding new Classifier as all classifiers agree on the same steps to predict the text, importing the model and classifying the text. We used the MVC design pattern in the back-end as MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

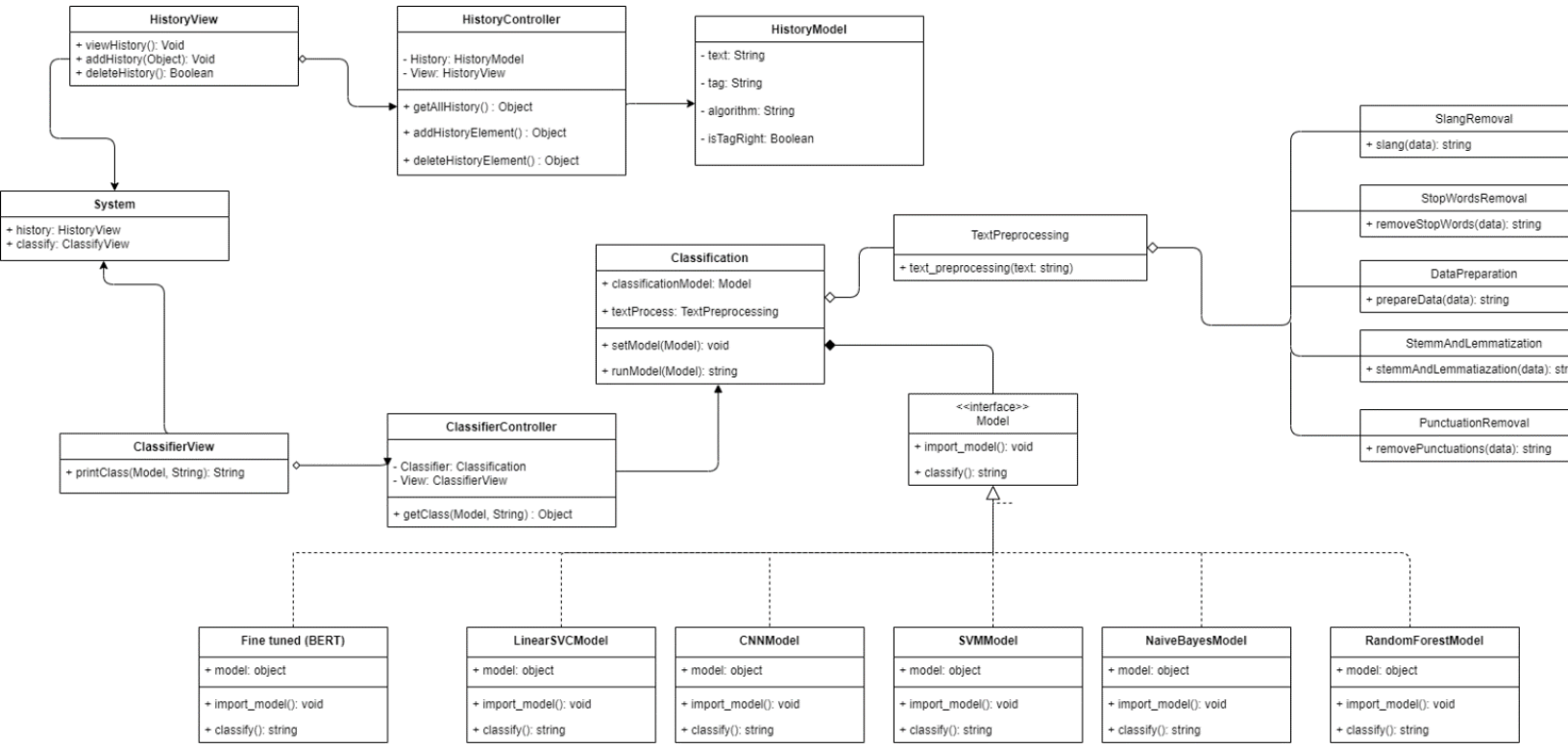

Figure 15 - Class Diagram

6.3. Sequence Diagram

As shown in Figure 16, the user sends a request with the text to Classification class the Text Preprocessing Class works on the text the classify the word and system add this classification process into history.

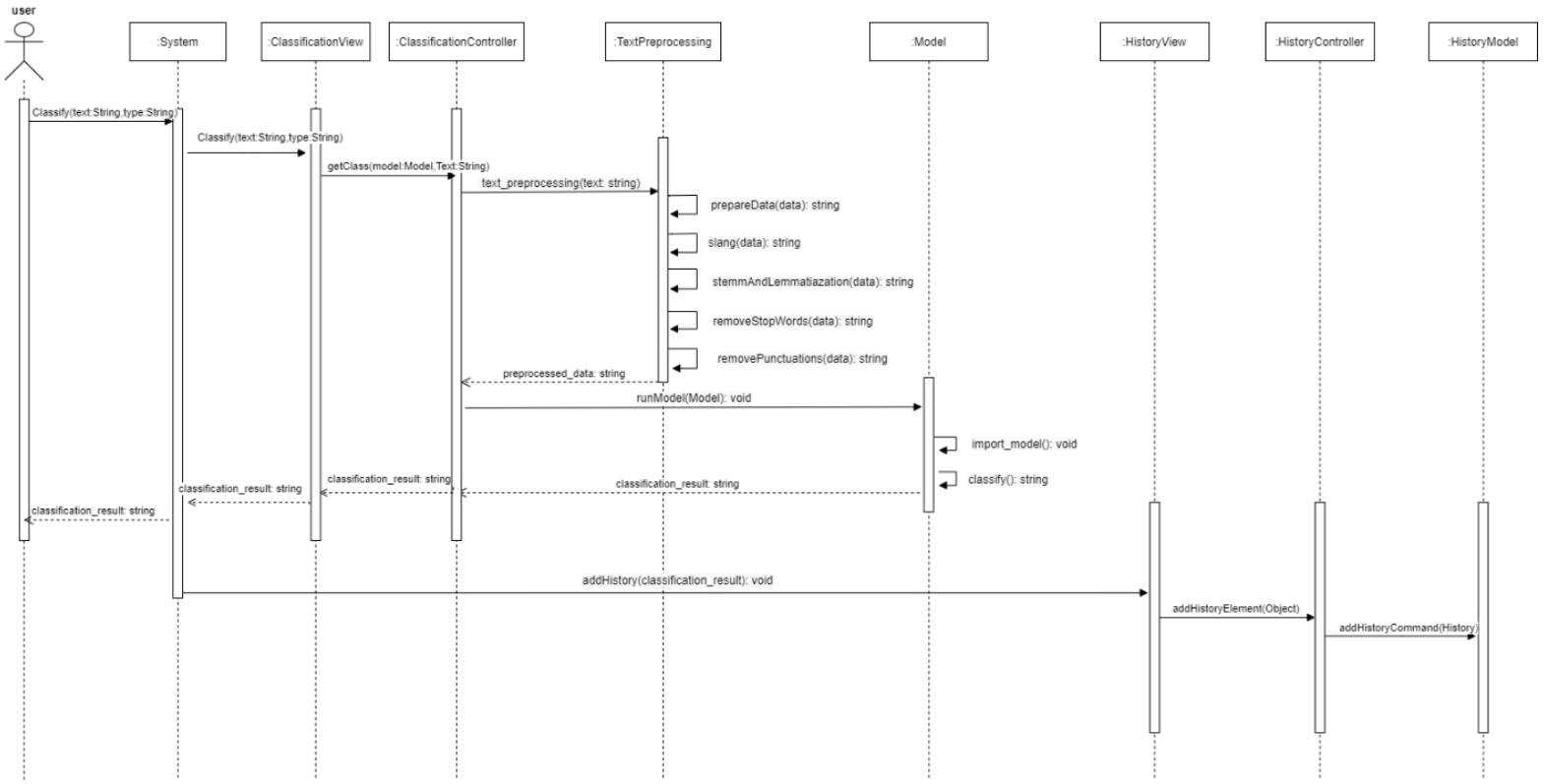

Figure 16 - Sequence Diagram - 1

As shown in Figure17, the user sends a request to system to view history and has an optional feature to delete a history element.

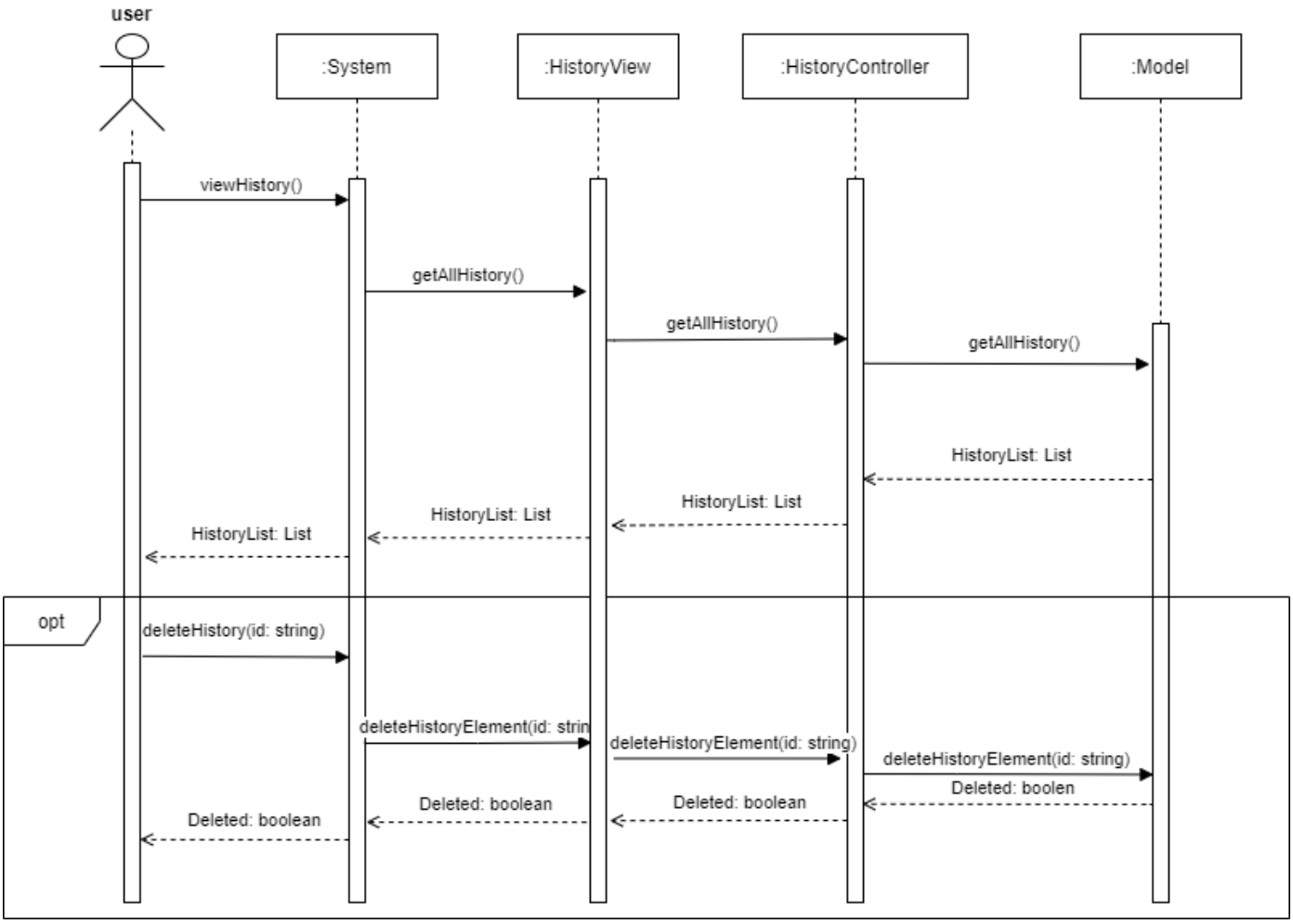

Figure 17 - Sequence Diagram – 2

6.4. Project ERD

As shown in figure 18, we have only one entity “history” with its attributes which user can add, remove, and get elements from this entity.

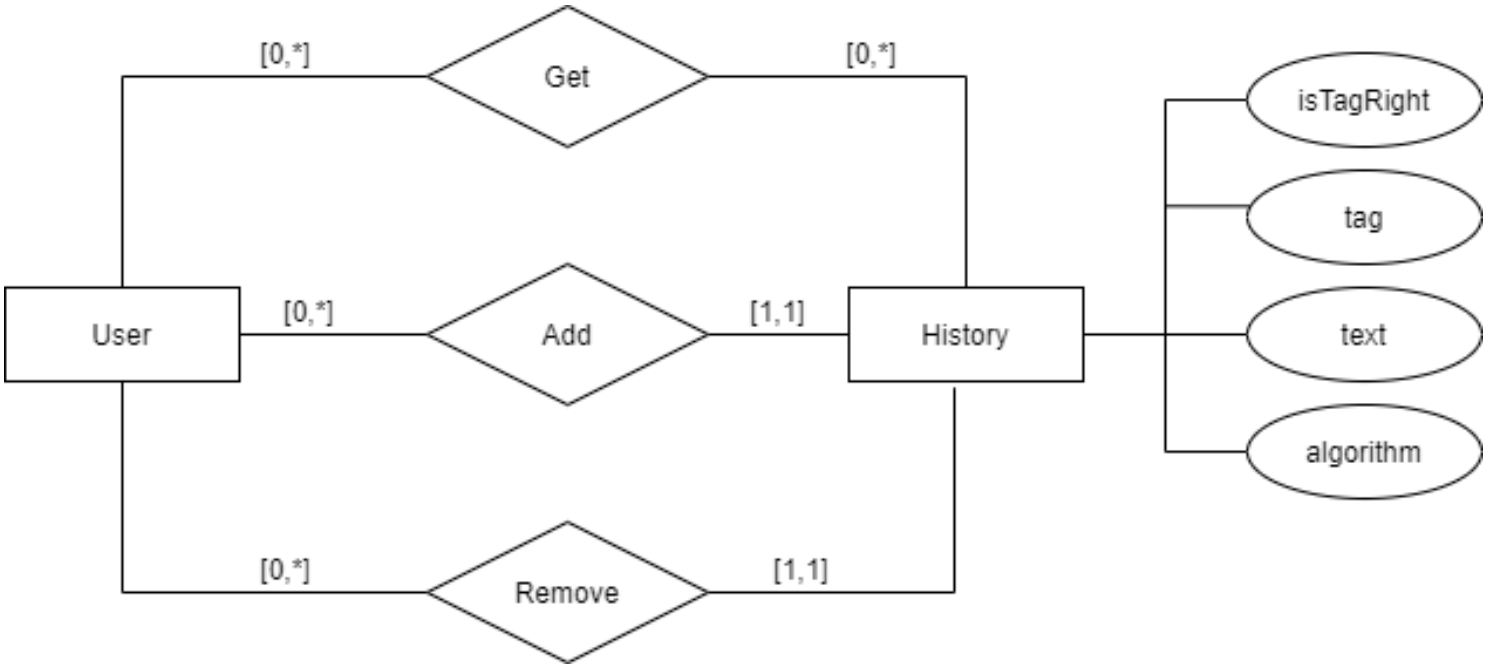

Figure 18 - Project ERD

6.5. System GUI Design

- Main Page: As shown in Figure 19
  - ➢ First field: where user can choose the algorithm.
  - ➢ Second field: where user can put the content to classify.

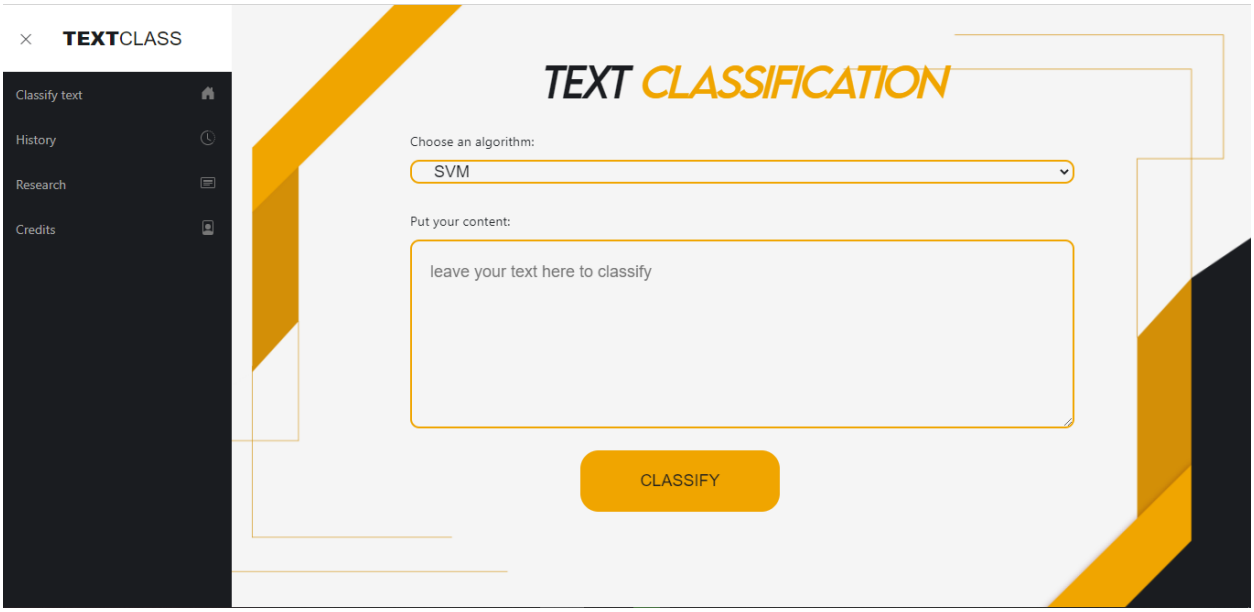

Figure 19 - GUI I

- History Page: As shown in Figure 20
  - ➢ Page where all previous classified content listed.

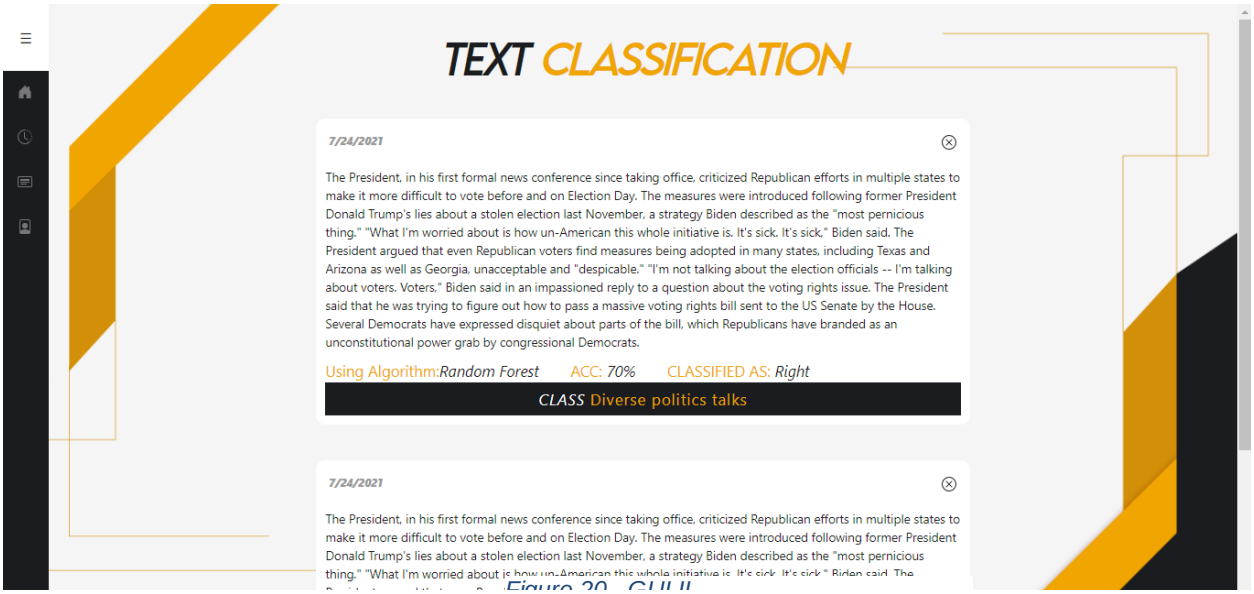

Figure 20 - GUI II

- Research Page: As shown in Figure 21
  - ➢ Page where we publish our research.

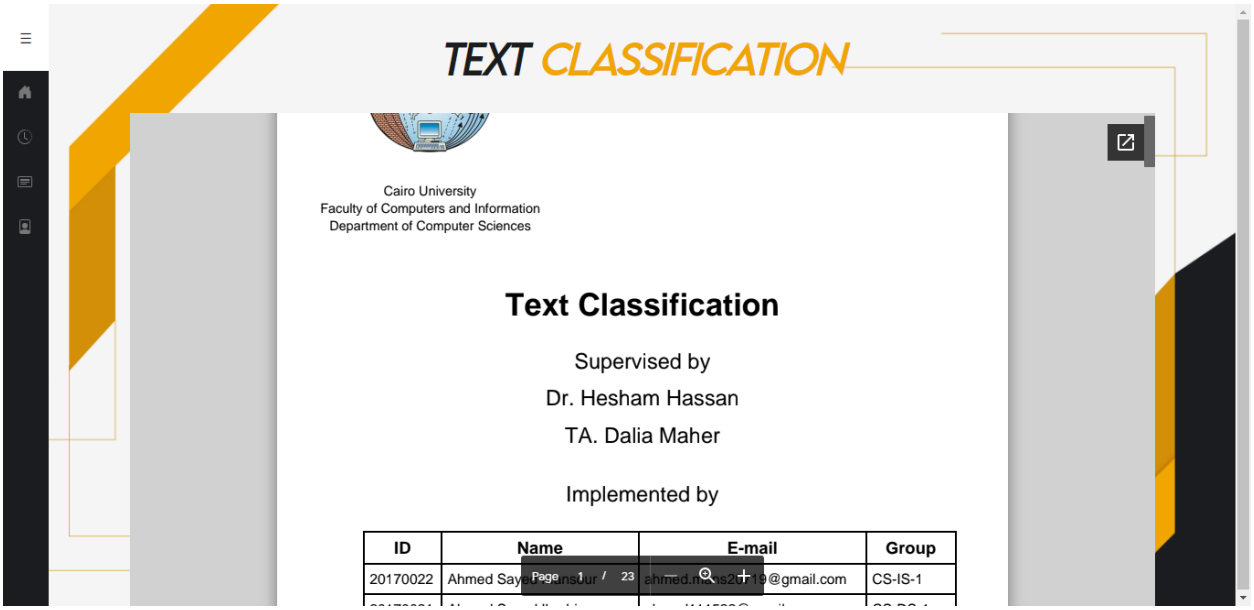

Figure 21 - GUI III

- Credit Page: Where all team contacts are listed. As shown in Figure 22

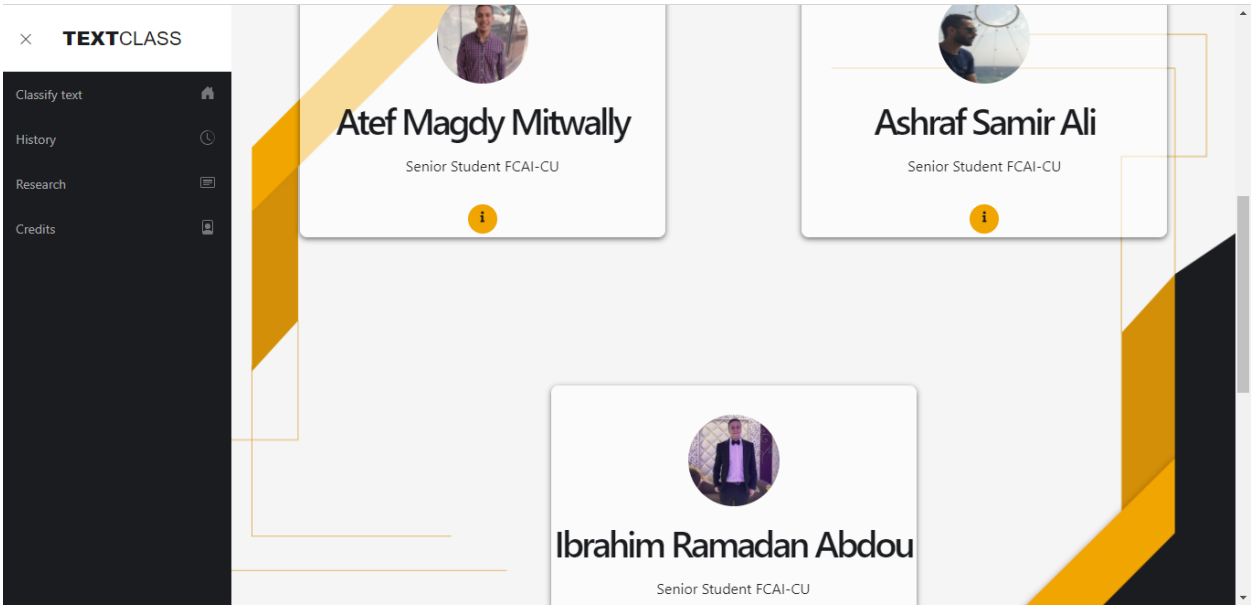

Figure 22 - GUI IV

7.0. Implementation and Testing

7.1. Models Results

After applying all steps and algorithms shown in the proposed text classification chapter, we compared all the models (as shown in figure 23) Accuracy on the 20newsgroups dataset

To find the best models to use on the **arxiv** dataset. We used (Naïve Bayes, SVM, Linear SVC, Random Forest, and CNN) the results as follows:

- Naïve Bayes : 66.67%
- SVM : 68.38%
- Linear SVC : 69.41%
- Random Forest: 57.17%
- CNN: 71.10%

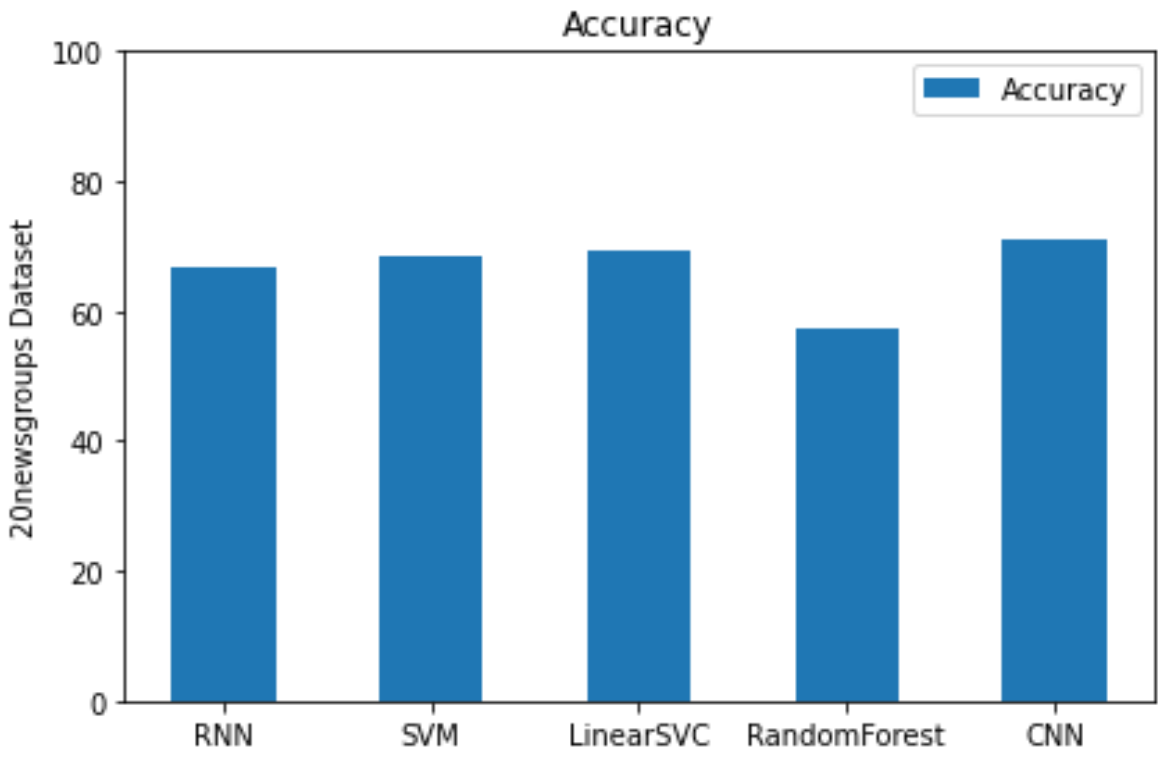

Figure 23 - Accuracy comparison I

After analyzing the results, we decided to use more efficient and complicated models like RNN with Pre-trained word embedding and Fine Tuning on BERT model. The results (as shown in figure 24) were:

- RNN with Pre-trained word embedding: 86.4%
- Fine tuning on BERT model: 76.5%

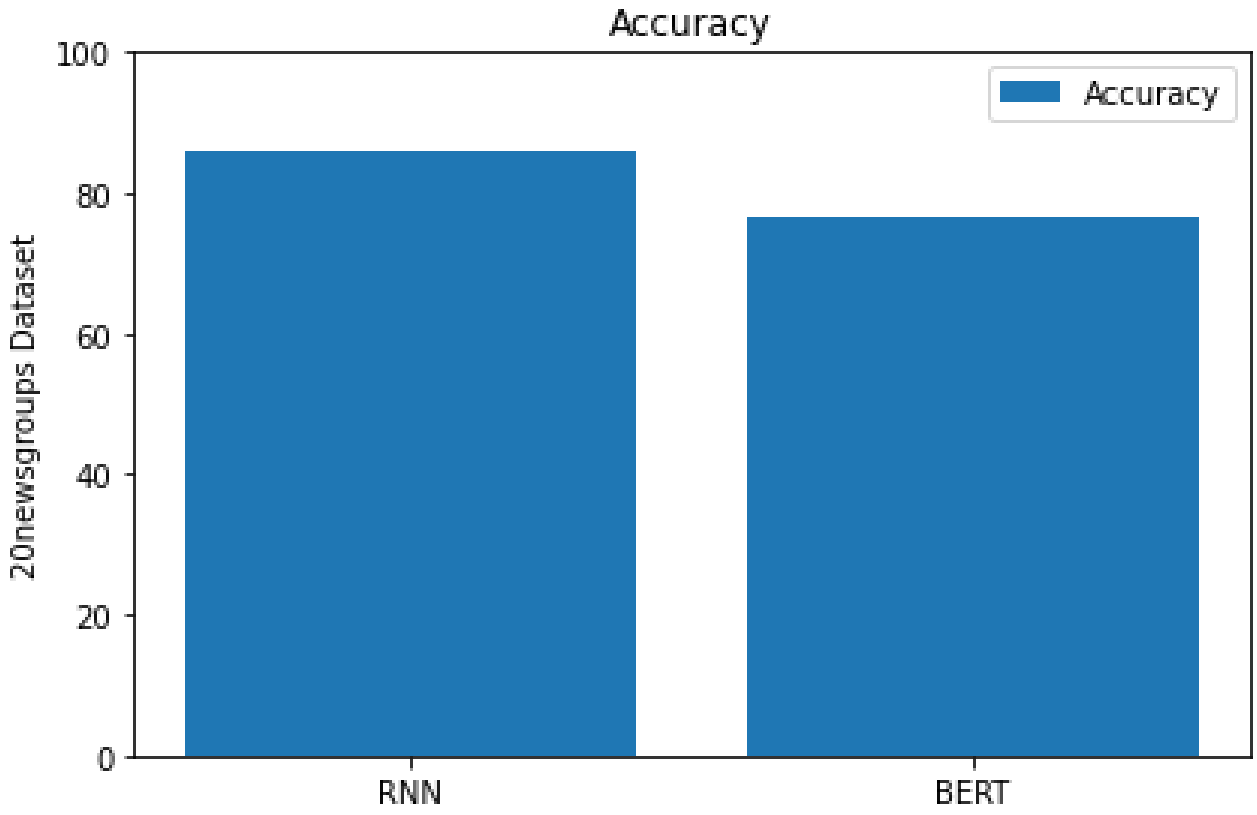

Figure 24 - Accuracy comparison II

We also tried to use arxiv dataset but we found it requires huge RAM and high GPU. As it contains more than 1500000 rows.

So we decided to use the LinearSVC model for this dataset with Results of 83.4% (as shown in figure 25) on the full arxiv dataset.

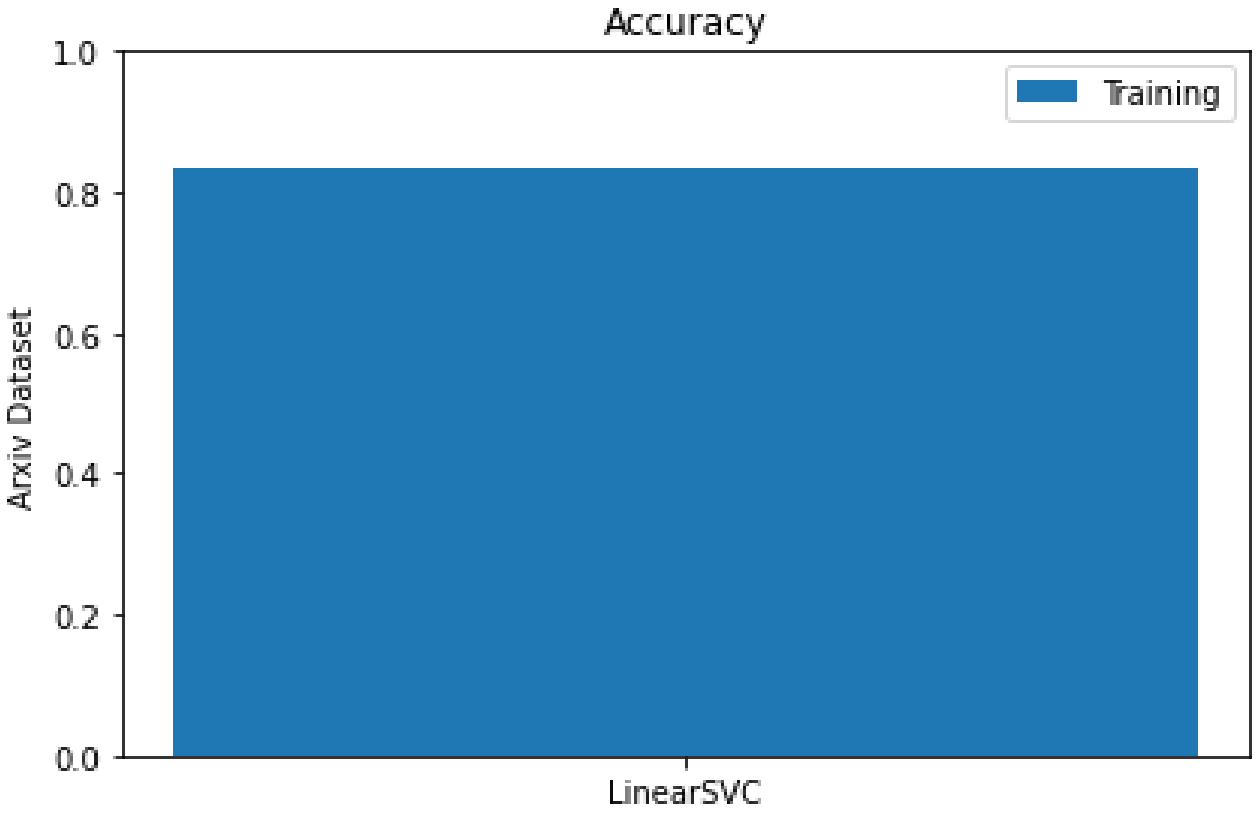

Figure 25 - Accuracy comparison III

7.2. Back-end

7.2.1. APIs

As shown in figure 26 our API mainly about classifying the document and the result for the remaining features.

Endpoints	Usage	Params
POST api/text/class	send the text to the model API then get the tag to send it to front end	`Request body` : { "text" : string, "option" : string} `Response body` : {"tag" : string}
GET api/classify	make user get the tag of a text	`Request body` : { "text" : string} `Response body` : {"tag" :string}
GET api/get/history	get all history from database	`Request body` : {} `Response body` : {"status" : 200, "history" :{}, "errors" : {}}
POST api/add/history	when user classify a text, a history is added to database	`Request body` : {"text": string, "tag": string, "algorithm": string, "isTagRight": true or false } `Response body` : {"status" : true or false, "historyId" :string, "errors" : {}}
POST api/delete/add	delete a history from database	`Request body` : {"id": string} `Response body` : {"status" : true or false, "historyId": string, "errors" : {} }

Figure 26 - API table

7.2.2. Server

In this project we have two servers

7.2.2.1. Flask

The flask server is mainly to handle the models and the prediction method. We mainly preferred Flask for its high performance with models (As shown in Figure 27).

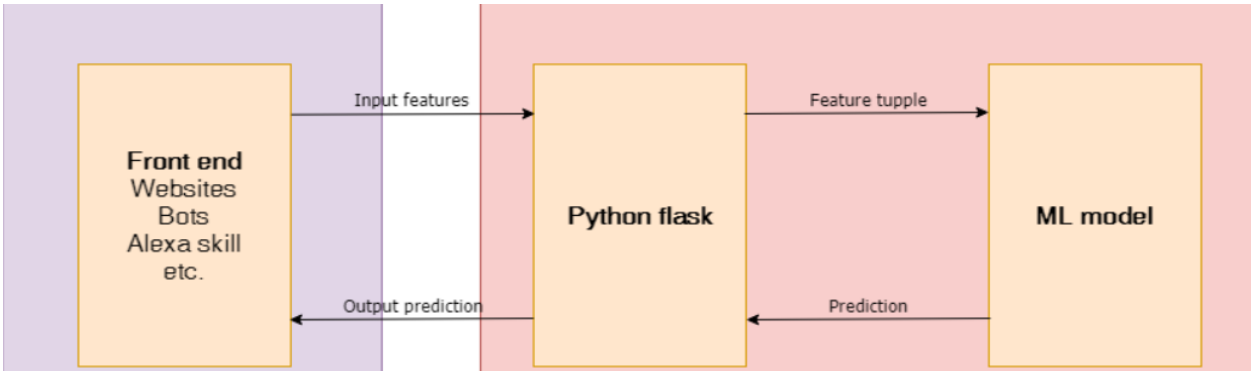

Figure 27 - Flask Architecture

7.2.2.2. Node JS

This server mainly communicates our back-end with front-end. This back-end is responsible for handling the front-end with desired data (As shown in Figure 28).

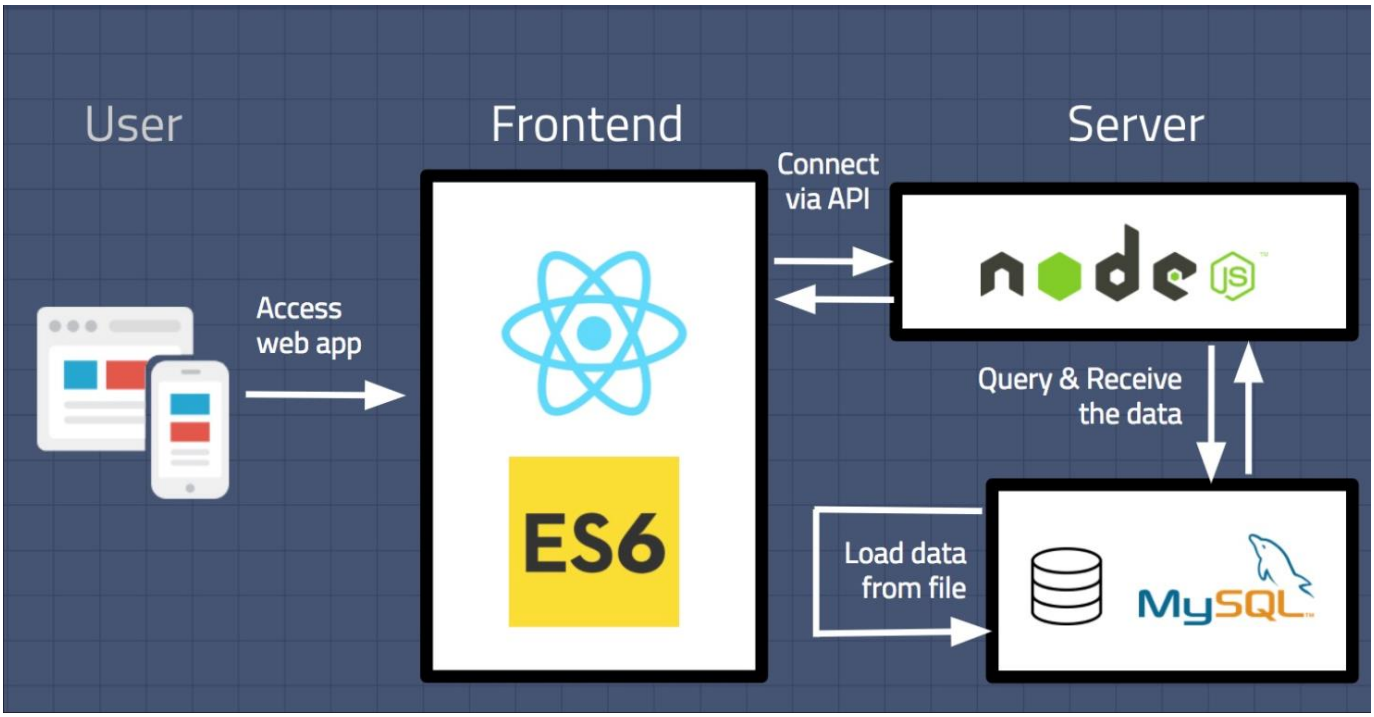

Figure 28 - Node JS Architecture

7.3. Test Cases

As shown in figure 29, test case using “Random forest” algorithm.

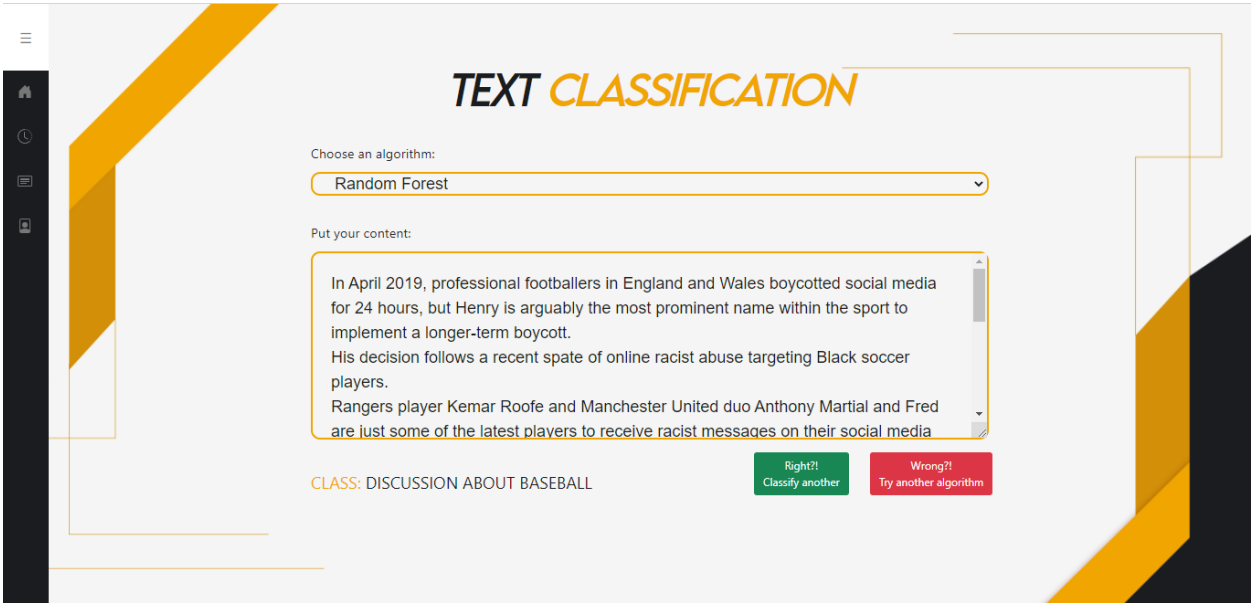

Figure 29 - Test Case 1

As shown in figure 30, test case using “SVM” algorithm.

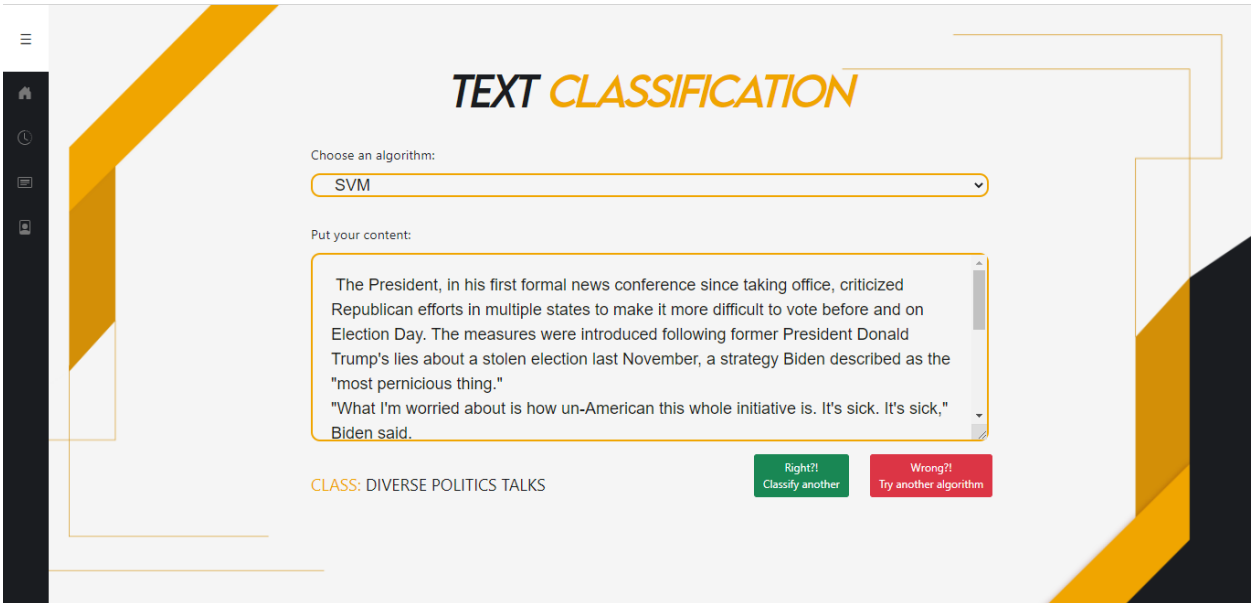

Figure 30 - Test Case 2

As shown in figure 31, test case using “Naïve Bayes” algorithm.

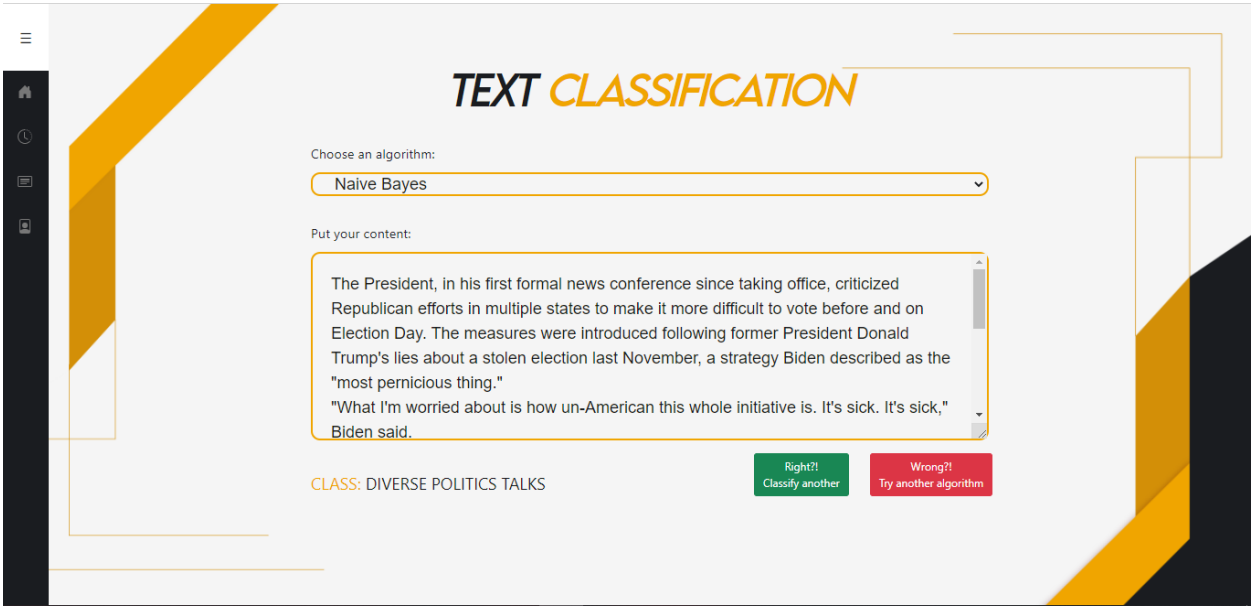

Figure 31 - Test Case 3

As shown in figure 32, another test case using “SVM” algorithm.

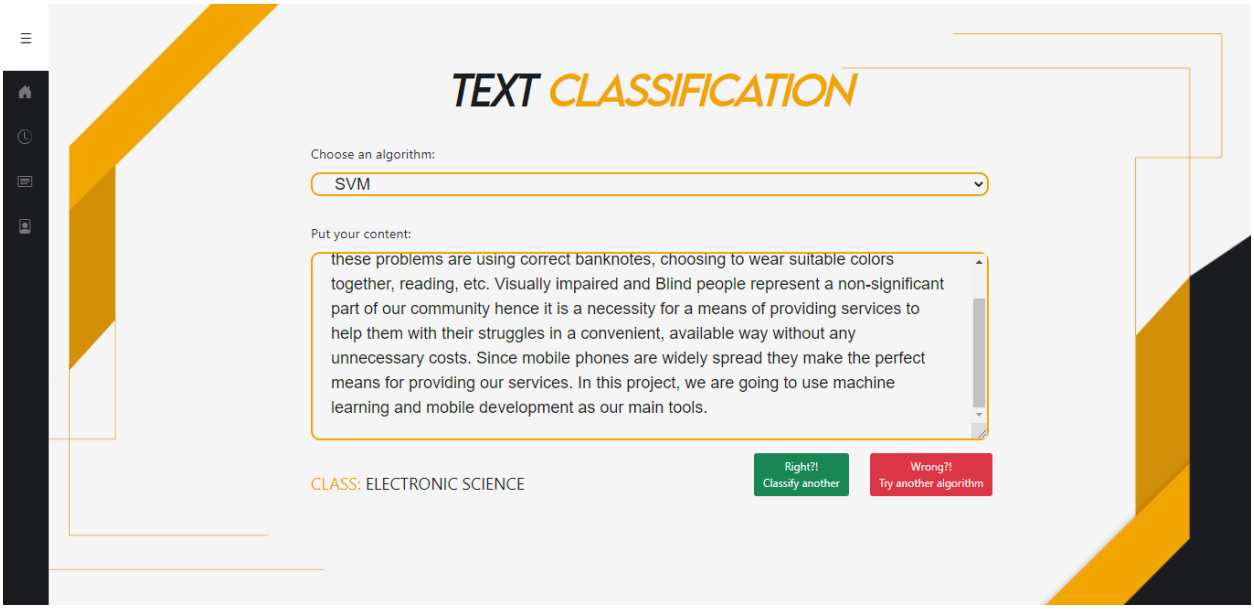

Figure 32 - Test Case 4

As shown in figure 33, test case using “Linear SVC” algorithm.

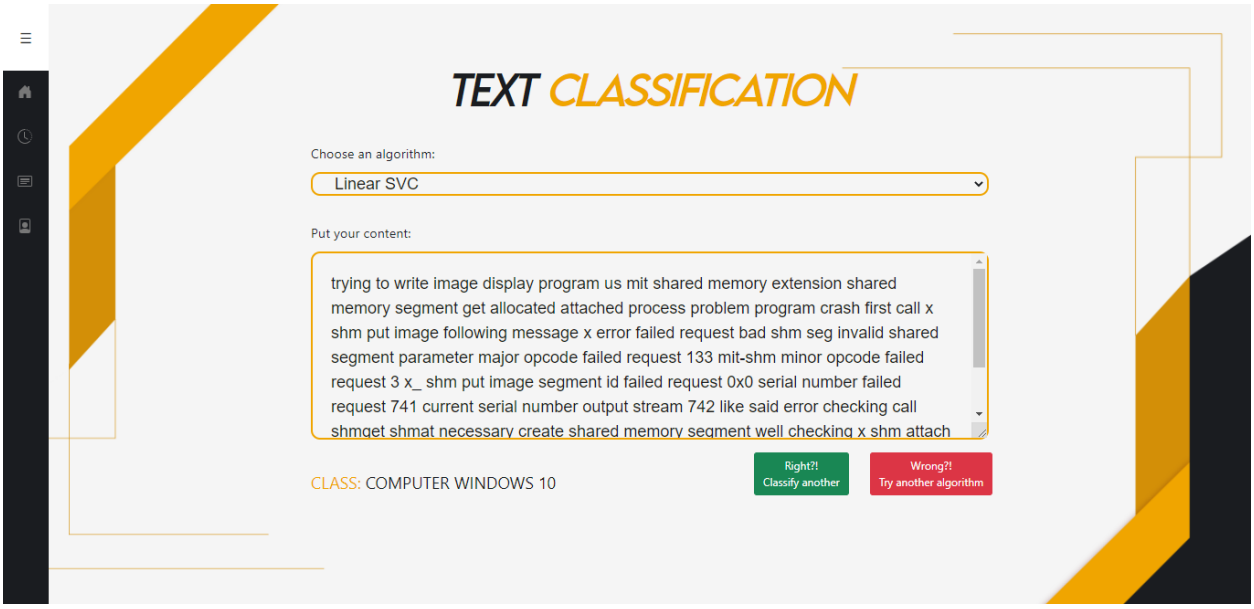

Figure 33 - Test Case 5

As shown in figure 34, another test case using “Linear SVC” algorithm.

Figure 34 - Test Case 6

As shown in figure 35, test case using “CNN” algorithm.

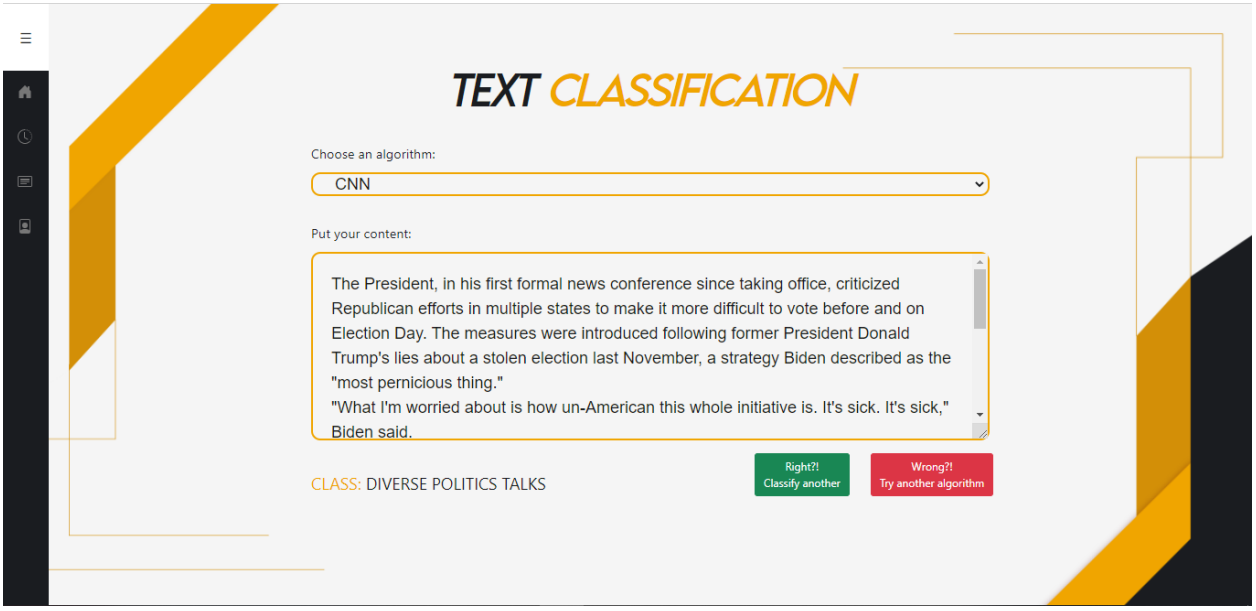

Figure 35 - Test Case 7

As shown in figure 36, test case using “Fine tune (BERT)” algorithm.

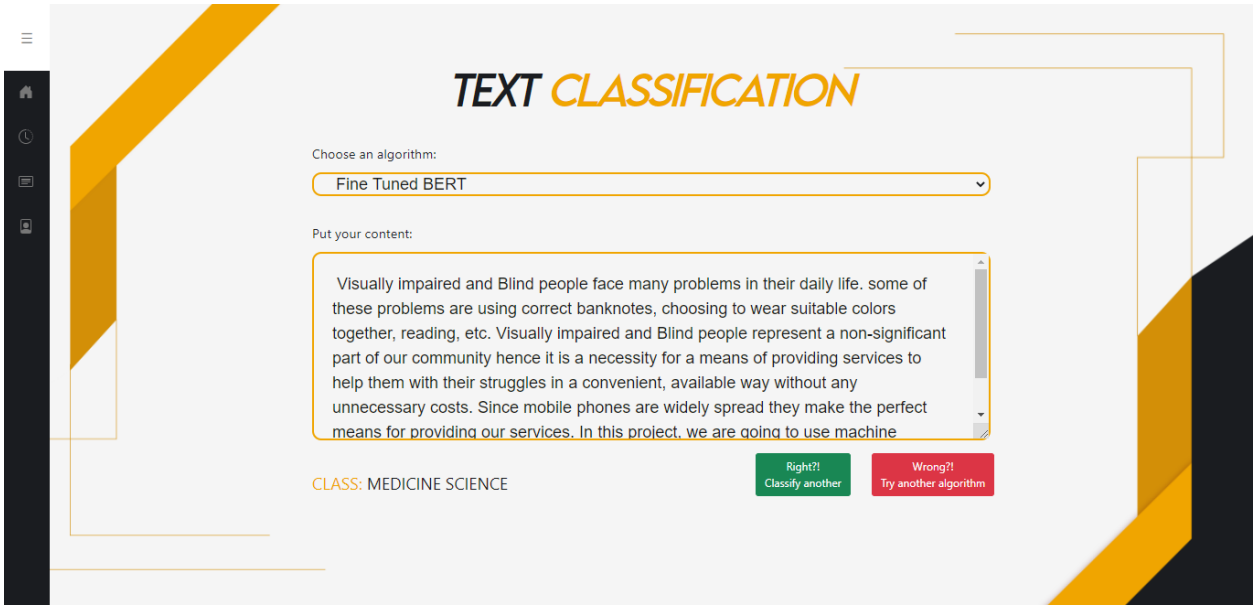

Figure 36 - Test Case 8

Conclusion

Text classification is a mature area of research by the increase of information flow available. It has seen large attention especially due to the high growth rate of the Internet and the importance of Internet search engines and generic classification of content on the Web. Process of text classification is well researched, but still many improvements can be made both to the feature preparation and to the classification engine itself to optimize the classification performance for a specific application. Research describing what adjustments should be made in specific situations is common, but a more generic framework is lacking. Effects of specific adjustments are also not well researched outside the original area of application. Due to these reasons, design of text classification systems is still more of an art than exact science.

References

[1] Monkeylearn.com. 2021. *Text Classification How text classification work*. [online] Available at: https://monkeylearn.com/text-classification/ .

[2] Monkeylearn.com. 2021. *Text Classification*. [online] Available at: https://monkeylearn.com/text-classification/ .

[3] Monkeylearn.com. 2021. *Text Classification Applications*. [online] Available at: https://monkeylearn.com/text-classification/ .

[4] Monkeylearn.com. 2021. *Text Classification*. [online] Available at: https://monkeylearn.com/text-classification/ .

[5] Monkeylearn.com. 2021. *Text Classification*. [online] Available at: https://monkeylearn.com/text-classification/ .

[6] Example, U., 2021. *Text Classification in Natural Language Processing*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example/ .

[7] Example, U., 2021. *Text Classification in Natural Language Processing*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example/ .

[8] Shaikh, J., 2017. Machine Learning, NLP: Text Classification using scikit-learn, python and NLTK... [Online] Medium. Available at: https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a .

[9] HUILGOL, P., 2021. Pre trained Models for Text Classification | Deep Learning Models. [Online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/03/6-pretrained-models-text-classification/ .