NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

GRADUATE PROGRAM
"COMPUTER SCIENCE"

MSc Thesis

Sentiment Analysis Application with Social Media Network Integration

Stavros D. Giannakis

Supervisor: **Christina Alexandris,** Associate Professor

ATHENS

SEPTEMBER 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΠΛΗΡΟΦΟΡΙΚΗ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή ανάλυσης συναισθημάτων με ενσωμάτωση μέσων κοινωνικής δικτύωσης

Σταύρος Δ. Γιαννάκης

Επιβλέπουσα: **Χριστίνα Αλεξανδρή,** Αναπληρώτρια Καθηγήτρια

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2020

MSc Thesis

Sentiment analysis application with social media network integration

Stavros D. Giannakis
S.N.: CS2180006

September 2020

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Ανάλυσης Συναισθημάτων με Ενσωμάτωση Μέσων Κοινωνικής Δικτύωσης

Σταύρος Δ. Γιαννάκης
Α.Μ.: CS2180006

Σεπτέμβριος 2020

ABSTRACT

Sentiment Analysis is an application domain of Natural Language Processing focusing in extracting sentiment and opinion from textual input. The purpose of the present Thesis is the creation of a web platform for extracting sentiment from texts. Specifically, the designed and implemented platform consists of a machine learning model that manages to retrieve sentiment from a text by categorizing the input into three different classes: positive, negative or neutral. Additionally, this model can be used via a web application processing two different types of text input, namely tweets and (movie) reviews. The platform also provides the possibility to retrieve comments and opinions from different social media platforms such as Reddit, Twitter and Youtube by searching any keyword and classify the results. The results are presented with a distinctive visualization to the users, giving a better perspective of what people think about specific topics. For the development of the components of the present project and application, the Python and JavaScript programming languages have been utilized. The machine learning model and the training data is described, as well as the preprocessing techniques that each textual input is subjected to before its classification into a category. Finally, improvements on the platform are proposed for offering more options and functionalities to the users.

SUBJECT AREA: Natural Language Processing, Machine Learning

KEYWORDS: Text Classification, Web Development, Sentiment Analysis, Opinion Mining

ΠΕΡΙΛΗΨΗ

Η εξαγωγή και ανάλυση συναισθήματος αποτελεί τομέα εφαρμογών της Επεξεργασίας Φυσικής Γλώσσας, στοχεύοντας στην εξαγωγή γνώμης και συναισθημάτων από εισερχόμενα κείμενα. Η παρούσα μελέτη και Διπλωματική Εργασία αυτή έχει ως σκοπό την δημιουργία μιας διαδικτυακής πλατφόρμας για εξαγωγή συναισθήματος από γραπτά κείμενα των μεσών κοινωνικής δικτύωσης. Συγκεκριμένα, αποτελείται από μοντέλο μηχανικής μάθησης το οποίο επιτρέπει την ανάλυση συναισθήματος από κείμενα και την κατηγοριοποίησή τους σε θετικά, αρνητικά ή ουδέτερα, καθώς και μιας διαδικτυακής εφαρμογής η οποία παρέχει σε χρήστες τη δυνατότητα να αλληλεπιδρούν με το μοντέλο αυτό και να εξάγουν συναισθηματική πληροφορία από τα κείμενα που εισάγουν. Επιπροσθέτως, δίνεται η δυνατότητα αναζήτησης λέξεων-κλειδιών σε δύο διαφορετικά είδη κειμένων (tweets και κριτικές) και σε διαφορετικά μέσα κοινωνικής δικτύωσης όπως είναι το Twitter και το Reddit, για την ανάλυση σχολίων, καθώς επίσης και η προβολή διαγραμμάτων με στοιχεία σχετικά με τους χρήστες που διατύπωσαν τα σχόλια. Δίνεται η δυνατότητα εξαγωγής σχολίων και ανάλυσης του συναισθηματικού περιεχομένου τους και από βίντεο της πλατφόρμας Youtube. Για την υλοποίηση των στοιχείων της εργασίας χρησιμοποιήθηκαν οι γλώσσες προγραμματισμού Python και JavaScript. Η παρούσα μελέτη περιλαμβάνει εκτενή περιγραφή του μοντέλου προβλέψεων και του τρόπου δημιουργίας του. Περιγράφεται επίσης αναλυτικά η υλοποίηση καθώς και η προ-επεξεργασία που υπόκειται το κείμενο πριν δοθεί στο μοντέλο μηχανικής μάθησης για την εξαγωγή συναισθήματος. Τέλος, προτείνονται βελτιώσεις και επεκτάσεις που θα μπορούσαν να γίνουν στην εν λόγω πλατφόρμα μελλοντικά, προσφέροντας ακόμη περισσότερες δυνατότητες και λειτουργίες στους χρήστες της εφαρμογής.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία Φυσικής Γλώσσας, Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Κατηγοριοποίηση κειμένου, Ανάπτυξη Εφαρμογών Παγκόσμιου Ιστού, Ανάλυση Συναισθήματος, Εξόρυξη Γνώμης

AKNOWLEDGMENTS

I would like to thank my supervisor, Professor Christina Alexandris, whose expertise was invaluable and her class, “Computational Linguistics”, provided me with the tools and ideas to successfully complete my Thesis.

CONTENTS

PROLOGUE 14

1. INTRODUCTION 15

1.1 Objectives 15

1.2 Content and Structure 15

2. RELATED WORK 17

3. THEORETICAL FRAMEWORK 18

3.1 Machine Learning 18

3.2 Computational Linguistics 19

3.3 Sentiment Analysis 21

4. PROJECT DESCRIPTION AND USE CASES 27

4.1 Outline of the System 27

4.2 Text type and Target User Group 28

4.3 Datasets 29

4.3.1 Sentiment140 Dataset 29

4.3.2 IMDB Dataset 33

4.4 Greek Data 35

5. PREDICTION MODEL 37

5.1 Technical Information 37

5.2 Data Preprocessing 37

5.3 Feature Extraction 38

5.3.1 Bag of Words 39

5.3.2 Term Frequency – Inverse Document Frequency 40

5.4 Classification Algorithms 41

5.4.1 Naïve Bayes 41

5.4.2 Logistic Regression 42

5.5 Training procedure 42
5.5.1 Normalization 43
5.5.2 Feature Extraction 44
5.5.3 Classification Algorithm 46
5.5.4 Combined Model 48

5.6 Evaluation 48

5.7 Summary 50

6. COMMUNICATION – DATA ACCESS LAYER 51

6.1 Technical Information 51

6.2 External API Communication 52
6.2.1 Twitter 52
6.2.2 Reddit 52
6.2.3 Youtube 53

6.3 API Endpoints 53
6.3.1 Free-text Endpoint 55
6.3.2 Twitter Endpoint 56
6.3.3 Reddit Endpoints 57
6.3.4 Youtube endpoint 60

6.4 Summary 62

6.5 Insights for Greek Data 62

7. USER INTERFACE – PRESENTATION LAYER 63

7.1 Technical Information 63

7.2 Homepage 64

7.3 Free-text Sentiment Analysis 65

7.4 Twitter 67

7.5 Reddit 68

7.6 Youtube 70

7.7 About 72

7.8 Summary 73

7.9 Adaptations for Greek Data 73

8. CONCLUSION AND FUTURE WORK 75

8.1 Conclusion 75

8.2 Future Work 76

LIST OF TRANSLATIONS 77

LIST OF ABBREVIATIONS 78

APPENDIX A. INSTALLATION GUIDE 79

REFERENCES 80

LIST OF FIGURES

Figure 1: Examples of Positive and Negative Words..21

Figure 2: Example Sentences...22

Figure 3: High level solution diagram 28

Figure 4: Wordcloud of the whole Sentiment140 dataset 30

Figure 5: Wordcloud of the positive words of theSentiment140 dataset 31

Figure 6: Wordcloud of the negative words of theSentiment140 dataset.......................31

Figure 7: Wordcloud of the whole IMDB dataset 33

Figure 8: Wordcloud of the positive words of the IMDB dataset 34

Figure 9: Wordcloud of the negative words of the IMDB dataset..................................34

Figure 10: chartData example 54

Figure 11: /api/text/ example request and response 56

Figure 12: /api/twitter/ example request and response 57

Figure 13: /api/redditSub/ example request...58

Figure 14: /api/redditSearch/ example request 59

Figure 15: /api/redditURL/ example request 59

Figure 16: Reddit endpoints example response 60

Figure 17: /api/youtube/ example request and response 61

Figure 18: Homepage layout 64

Figure 19: Extended sidebar..65

Figure 20: Free-text sentiment analysis layout 66

Figure 21: Free-text sentiment analysis with result...66

Figure 22: Twitter layout 67

Figure 23: Twitter results 68

Figure 24: Twitter detailed results..68

Figure 25: Reddit layout...69

Figure 26: Reddit results 70

Figure 27: Reddit detailed results 70

Figure 28: Youtube layout.. 71

Figure 29: Youtube results.. 72

Figure 30: Youtube detailed results 72

Figure 31: About layout 73

LIST OF TABLES

Table 1: Normalization results (sentiment140) 43

Table 2: Normalization results (IMDB)...44

Table 3: Count Vectorizer tuning results (sentiment140)..44

Table 4: Count Vectorizer tuning results (IMDB) 45

Table 5: Multinomial Naive Bayes tuning (sentiment140)...46

Table 6: Bernoulli Naive Bayes tuning (sentiment140)...46

Table 7: Logistic Regression tuning (sentiment140)...47

Table 8: Multinomial Naive Bayes tuning (IMDB) 47

Table 9: Bernoulli Naive Bayes tuning (IMDB)..47

Table 10: Logistic Regression tuning (IMDB) 47

Table 11: Combined dataset classifier scores 48

Table 12: Sentiment analysis examples 49

PROLOGUE

This thesis was written as a part of the MSc degree in Computer Science, offered by the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens.

1. INTRODUCTION

The recent involvement of Computer Science (CS) applications in everyday life and daily routines is becoming a necessity, providing solutions to practical problems with the automation and simplification of various tasks. With the advancement of "smart" machines and the communication between them and humans becoming easier due to recent cutting-edge technologies, it is necessary for systems to understand the human language and analyze it. Functions like Machine Translation (MT) and voice commands require that the microcomputers inside all applications and devices used in everyday life have the possibility to process, "understand" and parse the spoken or written words, to perform a specific action. The analysis of written and spoken language is considered a complex task as a result of the many factors that are related to language as input data. Recent Natural Language Processing (NLP) applications are not restricted in processing monolingual and multilingual language-specific linguistic features such as syntax and grammar and phonology, but target to process input data such as prosody and sentiment. NLP applications include various platforms that offer numerous types of text analysis, helping users retrieve essential information from textual input.

1.1 Objectives

The present project and application focuses in the sentiment analysis of written text input, a NLP application considered to be related to a set of interesting challenges. Extracting sentiment and emotion from a passage can prove to be valuable for a variety of reasons like recommendation systems and statistics of various reviews of specific items. The present project and application is intended for a broad range of users, including users who do not possess programming knowledge. It concerns a sentiment analysis application, namely a "machine learning model that understands the sentiment of a sentence". In particular, the application presented in this Thesis aims to provide a web interface for users (1) to analyze their sentences and (2) to evaluate batches of opinions and comments based on specific keywords, retrieved from not one but from various types of online sources. The online sources concerned include: (a) Twitter, a microblogging website, (b) Reddit, a social news aggregation, web content rating, and discussion website and, finally, (c) Youtube, an online video-sharing platform. The sentiment analysis of the input data will be based on custom machine learning models, trained by different datasets and based on probabilistic algorithms.

The advantages of the proposed project and application are the following:

(1) the user-friendly interface, providing an easy way for users to perform sentimental analysis on their input,

(2) the customized machine learning models,trained for two different types of text and document categories (tweets and (movie) reviews) and targeting to provide accurate results,

(3) the communication with multiple social media platforms for data retrieval based on specific topics.

1.2 Content and Structure

In the present Thesis, related research and projects in the area of Sentiment Analysis are described in Chapter 2. The connection of Sentiment Analysis to Computational Linguistics (CL) and Natural Language Processing (NLP) is presented in Chapter 3, as well as fundamental concepts and theories of Machine Learning (ML). The project and application with each respective basic component is presented and described in Chapters 4,5,6 and 7. Scenarios and use cases, as well as a brief description of the data and the target audience are presented in Chapter 4. Chapter 5 introduces the machine learning (ML) algorithms, data preprocessing, fine-tuning and evaluation of the final models. Chapter 6 explains the communication between the machine learning (ML) model and the interface of the application, as well as the communication with external APIs provided by the platforms integrated and the analysis of the textual data. Chapter 7 presents the User Interface (UI) of the application with a comprehensive description of the functionalities offered. Finally, Chapter 8 discusses further research including possible further upgrades of the application, such as a better user-experience and even more accurate results.

2. RELATED WORK

With the recent advances in Artificial Intelligence and the rising need of computers to understand the human language and it's components like syntax, grammar and emotion, Natural Language Processing (NLP) applications have been an attractive object of study for many research groups specializing in a variety of scientific fields. Artificial Intelligence researchers, computational linguists and cognitive scientists have presented a wide range of research and resources for a plethora of NLP tasks, with Sentiment Analysis considered one of the most challenging, since it concerns processing emotion in human language.

The growing use of Sentiment Analysis in the social media era is directly connected to the extensive use of social media platforms and the opportunity that individuals have to express their opinion on any matter. For instance, Sentiment Analysis on Twitter has been extensively analyzed in recent research, with Agarwal et al. [1] being a characteristic example and providing a basis for the present project and application. In particular, two systems are presented, a tree kernel model [1] and a feature-based model [1]. Both models are compared to a state-of-the-art unigram model [2]. Although both models outperform the previously proposed and discussed unigram model [2], it is the feature-based model that inspired the approach discussed in this thesis.

Colace et.al [3] present a probabilistic model with various algorithms and evaluate it on Twitter data. Furthermore, while Twitter is one of the best resources of people's opinions, product reviews have also been very important to the rise of Sentiment Analysis. The probabilistic model [3] that the presented system utilizes was a major influence on the prediction models behind the application presented in this Thesis, since the approach selected was probabilistic as well.

It should be noted that Twitter has been the basis of Sentiment Analysis research and one of the largest data sources, with many researchers making it the focus of their studies. Go et.al [4] present a complete Machine Learning-based Sentiment Analysis project using a variety of algorithms, comparing them and discussing the results. Of course, social media in general can offer significant datasets for Sentiment Analysis. Troussas et.al [5] focus on textual data retrieved from Facebook and Paltoglou et.al [6] turn their focus and direction to Myspace and Digg in their research.

Yasen et al. [7] discuss the classification of movie reviews. Reviews in general tend to be more structured than the 140-character Twitter posts, which affect the classifier parameters and, sometimes, the algorithm selection. For the present project and application, this resulted to the decision to include multiple datasets during the training procedure, in order to classify different types of textual data more accurately and provide the best possible results.

Additionally, since the present Thesis targets to combine the machine learning model with an easy to use web interface, it relies to a remarkable extent to the approach of Kasper et al. [8], which concerns a platform using a crawler to collect hotel reviews and analyze them for the end-user. In this approach [8], the user interface places a priority in explaining the results to the user. The approach of Kasper et al. [8] is, therefore, one of the major components of the present project and application, aiming to develop a modern and responsive user interface to visualize the results for the users.

3. THEORETICAL FRAMEWORK

With the recent advances in Artificial Intelligence and technology in general, machines become "smarter", offering certain functionalities that were not expected a decade or even few years ago. With systems of interrelated computing devices like the Internet of Things (IOT), microprocessors reside in almost every machine, including devices in households; fridges, smart vacuums, interior and exterior lighting systems are a few of the wide variety of devices and tools that have become "smarter. Of course, the human factor still plays a major role, organizing and adjusting the settings the appliances to perform in a particular manner. "Smarter" machines have resulted to an increased need for better communication between computers and humans and, subsequently, make the demand for computers ability to understand the human language essential. While parsing voice commands and understanding simple instructions is challenging, it does not come close to the levels of difficulty of more complex tasks in Natural Language Processing (NLP) and Computational Linguistics. Natural Language Processing (NLP) includes tasks as varied as Part-of-Speech tagging, contextual machine translation and extraction of emotion. Sentiment Analysis is considered one of the more recent Computational Linguistics and NLP applications and is also related to Machine Learning.

3.1 Machine Learning

Artificial Intelligence (AI) is intelligence demonstrated by machines, which is differentiated from natural intelligence displayed by humans and animals. For intelligent machines to exist, researchers realized that these machines should learn from their environment, adapt to new conditions and improve based on continuous input. Therefore, Machine Learning is a field of Artificial Intelligence where computer algorithms are used to autonomously learn from data and information, improving their algorithms by themselves, without human interaction [9]. Machine Learning has a truly multidisciplinary character, incorporating ideas and basic concepts from diverse fields, such as Statistics, Computer Science, Information Theory and other fields and disciplines, which is also a reason for its rapid growth the last decade; it is the focus of many researchers [10]. The data that exists in the world is growing exponentially, resulting in the ability to train machine learning models for different applications. Functions like face and object recognition from images [11], contextual machine translation [12] have been improved significantly, yielding accurate results with significant speed with the aid of rising computational power in modern computer processors. However, there is currently no system that can do all the required work in a single and specific task. For example, there is no machine learning model that can be used to translate every language into all the others, or identify thousands of different objects. Divided in smaller tasks, like the translation between two languages or the distinction between a few objects, machine learning models have been successful and maybe, in the near future, systems that can identify objects between thousands of categories or translate between a variety of languages will be created.

Machine learning algorithms build a mathematical model based on sample data, in order to make predictions on various subjects. Depending on the goal that each system aims to achieve, there are plenty of machine learning techniques and algorithms that can be utilized. We will list a few of these approaches, combined with some example use-cases to provide a clearer image. It should be noted that some of the examples given below can be achieved using multiple techniques and routines.

1. Classification: Given an input, the algorithm categorizes it into one of the available categories. Classification problems can be either binary or multi-

class. The Classification technique can be used for image classification [13], sentiment analysis [14] and various tasks where we need to categorize a textual input like fraudulent detection. Some of the classification algorithms are Logistic Regression, Support Vector Machines (SVM) and Naïve Bayes family.

2. Clustering: Groups related items together into various clusters, based on their features. Clustering algorithms such as K-Means and Singular-Value Decomposition (SVD) are commonly used in recommendation systems and targeted marketing applications, where decisions have to be made based on the items categorized on the same cluster.
3. Dimensionality Reduction: The technique that transforms data from a high-dimensional space into a low-dimensional representation that contains valuable information and properties of the original data. This helps with the manipulation of the data, the removal of noise and invaluable features and the reduction of computational times and costs. One of the most popular algorithms is Principal Component Analysis (PCA) and is used for face and image recognition, as well as text mining [15].

All the above-mentioned machine learning techniques, including others that are not presented here, can be categorized by their learning approach; they differ in the data they receive as input and serve as output and the way they function during training. We will discuss the three (3) most common categories of learning approaches:

1. Supervised learning: These kinds of algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The user feeds training examples labeled with the correct class and the algorithm transforms the data into vector representations. Then, given an unlabeled input, the algorithm will try to predict the correct label. The classification problem requires a supervised algorithm.
2. Unsupervised learning: In contrast with their supervised counterparts, unsupervised algorithms do not require any target input from the developer. Given a dataset, the algorithms examine the structure of the data and compare new data additions based on the presence or absence of commonalities of the existing data. Clustering techniques work with unsupervised algorithms.
3. Semi-supervised learning: A combination of supervised and unsupervised learning. The semi-supervised learning approach uses a small amount of labeled data, as well as a larger amount of unlabeled data during training. Semi-supervised learning was created to counter the disadvantages of the first two learning methods: the costly process of labeling huge amounts of data of supervised learning as well as the limited application spectrum of the unsupervised counterpart. With the semi-supervised learning approach, the unlabeled data is clustered and then the existing labeled data is used to predict the classes of the entries that are not labeled.

3.2 Computational Linguistics

Computer Science in general is no stranger to being a part of interdisciplinary fields, including Linguistics. Computational Linguistics combines the methods and techniques of Computer Science with the theoretical models of Linguistics. With the recent advances in statistical models and neural networks, Computational Linguistics also explores natural language from a computational perspective. Computational Linguistics covers a wide and varied range of applications that can be divided into various categories. For example, one categorization depends on whether the input is written or spoken and the respective Text Processing and Speech Processing applications. Speech Processing applications may concern processing spoken input from Speech Recognition ASR systems or modelling produced spoken output from Speech Synthesis (TTS) Systems, in some cases in a dialogue setting (Spoken Dialog Systems). Computational Linguistics concerning Speech Processing may also generally explore how human language can be automatically processed and interpreted. [16]

The field of Computational Linguistics is extensive because the natural language is important in any application [17]. Typical applications are the following:

1. Machine Translation (MT): Since there are thousands of languages in the world, all with diverse features such as grammar and vocabulary, Machine Translation is considered a complex and challenging task. Machine Translation is an automated process within which computer software is used to convert text from one natural language to another. This process demands an expertise in grammar, structure of sentence and its meanings in the source and target languages. [18] [19]
2. Information Retrieval (IR): Information Retrieval refers to the process of obtaining information from natural language through text, usually within a specific context (user query).
3. Human-Computer Interaction (HCI): Applications that assist with the communication between humans and computers or any other electronic devices in natural language fall under this category. Automatic sentence completion, speech to text and vice versa, as well as voice commands are prime examples of Human-Computer Interaction functionalities that are within the scope of Computational Linguistics.

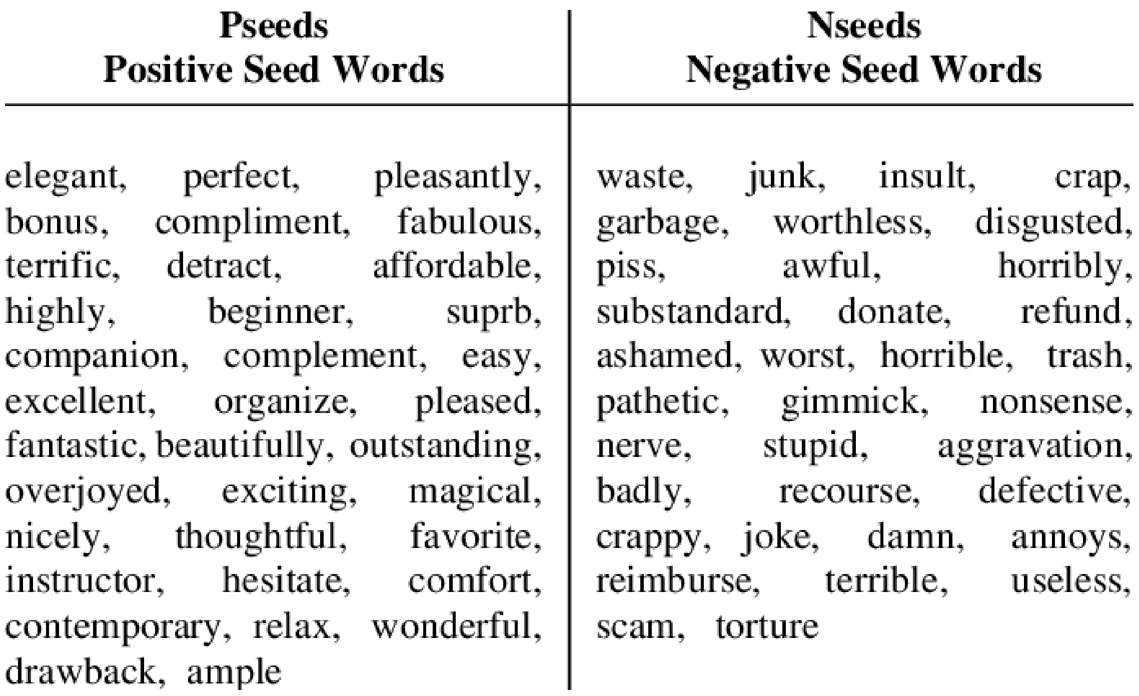
Computational Linguistics and Natural Language Processing (NLP) also include fields and applications that have a closer connection to Computer Science than Linguistics, such as Machine Learning. Usually, computer-based linguistics was performed by computer scientists who specialized in the field, also known as Natural Language Processing (NLP). Nowadays, with the evolution and progress of NLP tasks, many other fields of science cooperate for the optimal results, such as Statistics, Psychology and Neuroscience. If a definition of Computational Linguistics had to be created, it would be the scientific field that studies language from a computational perspective and is interested in providing computational models for various kinds of linguistic phenomena. [20]

Linguistics, in general, studies the meaning, context and form of the language, making the field extremely broad and diverse. Despite the recent advances in Computational Linguistics, Computer Science and Artificial Intelligence, the task of creating machines that understand natural human language, either spoken or written remains a challenge. Many obstacles stand in the way: From the variations in natural languages and dialects in grammar and syntax to variations and nuances in spoken languages, to the complexity of discourse processing and Sentiment Analysis. These are only a few of the issues that occur when textual information is the input in a machine. To deal with this complex input and its complications, computer scientists have used various methods like Machine Learning and Deep Learning, bringing in the equation other scientific fields too, like Statistics. While a significant number of issues occurring in Natural Language Processing have been dealt with, many others are too complex to train a computer to understand them completely, like sarcasm and ambiguities leading to wrong predictions and understanding from the machine’s perspective. An example of these issues is context. Context being the circumstances that form the setting for an event or conversation, it plays a major role in understanding the true meaning of a sentence and extract valuable information from it. With the knowledge of context, information on the emotional polarity of a sentence can be gained, resulting into better understanding of a situation. The definition and processing of context constitutes an essential yet challenging task in a variety of Computational Linguistics and Natural Language Processing systems, ranging from discourse processing applications to Sentiment Analysis.

3.3 Sentiment Analysis

Sentiment Analysis (SA) is the usage of various techniques derived from Natural Language Processing, Linguistics, Statistics and various fields that are related to Computational Linguistics, whose goal is to extract sentiment, emotion and opinion that is expressed in a text. In general terms, the scope of Sentiment Analysis is to determine the attitude, view or judgment of a person concerning some specific topic or the overall contextual polarity of a document. Sentiment Analysis is considered a field of Natural Language Processing, considered one of the most challenging NLP applications due to the many factors that are related to emotion as information content of textual input.

Sentiment Analysis is regularly referred to as Opinion Mining (OM) which is an almost synonymous term and these definition names are used interchangeably, with a small window of them referring to slightly different things. Sentiment Analysis focuses solely on the emotional aspect of a sentence, whereas Opinion Mining concentrates on the author’s opinion [21].

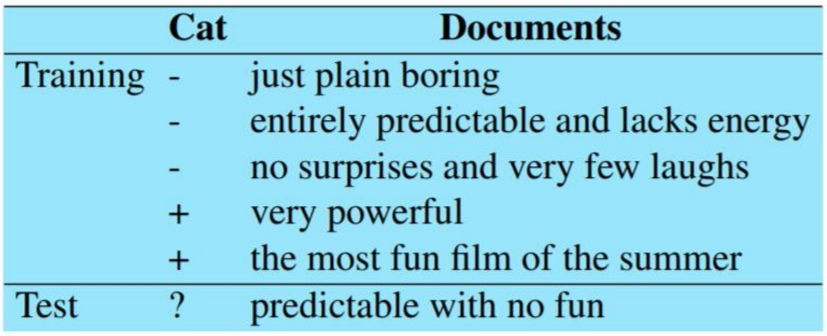


Pseeds Positive Seed Words	Nseeds Negative Seed Words
elegant, perfect, pleasantly, bonus, compliment, fabulous, terrific, detract, affordable, highly, beginner, suprb, companion, complement, easy, excellent, organize, pleased, fantastic, beautifully, outstanding, overjoyed, exciting, magical, nicely, thoughtful, favorite, instructor, hesitate, comfort, contemporary, relax, wonderful, drawback, ample	waste, junk, insult, crap, garbage, worthless, disgusted, piss, awful, horribly, substandard, donate, refund, ashamed, worst, horrible, trash, pathetic, gimmick, nonsense, nerve, stupid, aggravation, badly, recourse, defective, crappy, joke, damn, annoys, reimburse, terrible, useless, scam, torture

Figure 1: Examples of Positive and Negative Words. Reprinted from *Unsupervised Genre-Based Multidomain Sentiment Lexicon Learning Using Corpus-Generated Polarity Seed Words* [22]

Most of the Sentiment Analysis and Opinion Mining applications that exist nowadays are focused on labeling the polarity of a text as either a set of classes (for example: positive, negative, neutral) or a number that represents the that the document is positive or negative. Of course, developers can easily interfere with this and format their result in different ways like grading the results. A couple of examples of emotion extraction from texts are the following:

1. "I really enjoyed this conference, thought it provided the audience with great detail about the subject": These sentences hold a positive emotion. Reading it makes the user feel that the author had a pleasant time at the event he attended [23].
2. "The service at the restaurant was the worst. The food reached our table cold and the waiters were indescribably rude.": Opposite to the first example, the second expresses a negative sentiment [23].



	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Figure 2: Example Sentences. Reprinted from *Speech and Language Processing* [24]

While the examples above may seem may be considered a rather straightforward task and simple from a human's perspective, computers have a harder time understanding emotion. Later in this section, a few details about the way sentiment analysis works will be provided, assisting in understanding the examples above from a machine's standpoint. Moreover, challenging cases that lead to mistakes and difficulties in the sentiment analysis process will be presented and compared to the previously listed examples.

To give a more thorough understanding of sentiment in a document, the main components of an opinion expressed in a document are [25]:

1. Opinion holder: The person that holds an opinion or feeling on a particular object.
2. Entity: The object, that the opinion holder is referring to.
3. Opinion: the view of the opinion holder on the entity. Can have positive, negative or neutral orientation.

Of course, since documents are lists of sentences, several different opinions can exist in the same document.

While the area of Sentiment Analysis recently enjoyed a burst of research activity, there was always interest in the field, with one of the earliest works regarding belief models was published as a thesis in 1979, titled "Subjective understanding: Computer models of belief systems" by Jaime Guillermo Carbonell [26]. Carbonell's thesis sparked an interest in the subjective analysis topic, leading to one of the first books published about it in 1981 [27]. Unfortunately, the book does not include any technical information about the development environments or computers used back in the day. Towards the end of the 90s decade, computer intelligence was starting to rise, leading to more and more research and experiments in Natural Language Processing and Sentiment Analysis in particular and the publication of a significant number of research papers [28].

During the early days of the World Wide Web, it was difficult for the end-user to produce content, even if that content was just textual. The information residing on the internet was mostly generated and published by organizations that owned web pages, with no personal content existing whatsoever. With the introduction of Web 2.0 platforms in the 2000s, there was a noteworthy change of scene, with forums, blogs and online social networks providing users a podium to express their opinions and thoughts on specific matters.

This explosive growth in the field of Sentiment Analysis continued in the new millennium, with important steps on the field being made [29] [23] and setting the foundation for future work. With the beginning of the social media age, Sentiment Analysis became more and more the focus of computational linguists and other scientists, with people expressing their opinions and feeling all around the World Wide Web. Microblogging services like Twitter make it easy for developers to gather massive amounts of textual data containing people’s opinions and apply sentiment analysis techniques to extract information. These messages that people post on Twitter are real time and usually discuss current events, thus making the analysis of considerable importance [1]. The exponential growth of available information in the Web, in combination with the advanced technology and the possibility to use the internet almost everywhere and from a variety of machines, results to the exponential growth of the quantity of subjective information. This has provided new opportunities for linguists, in cooperation with computer scientists and data analysts, to create applications that extract sentiment and opinions from textual inputs.

Considering all the above, it is evident that the amount of information that can be collected using Sentiment Analysis and its more specific domains like Opinion Mining is considerably large, making it one of the most interesting parts of Natural Language Processing. Sentiment Analysis is not only of interest for theoretical work but also for practical applications. Sentiment analysis is considered to be one of the most rapidly growing research areas of NLP and Computer Science in general, with more than 7000 papers published after 2004. [30]

One of the greatest questions regarding the subject, is “Why Sentiment Analysis?”. Is Sentiment Analysis worth this effort and research? Are there any significant benefits and use cases for scientists to invest in this field for almost half a century? According to the available facts and information, the answer is yes. Sentiment analysis and Opinion Mining is connected to significant advantages and goals, namely the processing of large datasets with objective criteria (1), the efficient processing of customer opinions and feed-back (2), the speed of data analysis (3) and the quick and efficient response to user or customer input (4) [31].

Processing large datasets with objective criteria (1): A generic benefit is the rapid analysis and extraction of verdicts on huge datasets. Reducing the effort of people working on data analysis by assisting them with sentiment analysis models is a plus. Since people are not consistent on the criteria that they decide on the sentiment of a text, tagging text by sentiment is highly subjective, influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.

Efficient processing of customer opinions and feed-back (2): One of the most successful usages of Sentiment Analysis in the recent years is the ability to monitor customers opinions and thoughts about specific brands, products or services. Business owners very cost-effectively examine the thoughts of people regarding a recent product or service release, complaints that they may have and examine the success rate of their newly launched product. This procedure, of course, can also be used on older releases and help with the improvement and rejuvenation of a specific product line. Another advantage of using Sentiment Analysis for customer opinions and feed-back is the ability to act swiftly by investigating the root cause of a problem if a sentiment analysis model makes any detection over customers unfavorable remarks.

Data analysis speed (3): Newly created marketing strategies can be created and tested in a rapid fashion, due to the quick analysis of the data, that when done manually, might take many more days to finalize and extract conclusions from. With the introduction of computers, the analysis phase is multiple times faster, leaving time, space and resources for other teams to focus on their product strategies.

Quick and efficient response to user or customer input (4): Automated sentiment analysis can assist customer service with starting a conversation with customers that either are dissatisfied or happy with their product, by analyzing their email or comment on a social media page and replying swiftly with the appropriate message. For example, if someone expresses their disappointment of a product on Twitter or another social network, by searching the product in question and finding that specific comment, an automated response can be issued. Furthermore, this type of actions show commitment from the side of a business, earning the respect and trust of customers, even those that had a negative experience before.

Although Sentiment Analysis is connected to a broad spectrum of scientific fields, including Linguistics, and there are many ways sentiment analysis issues can be dealt with from a computer scientist’s perspective, the main approaches to Sentiment Analysis can be divided into two major categories: Lexicon-based (or Knowledge-based) [32] [33] and Machine Learning.

The Lexicon-based procedure involves calculating semantic orientation of words, sentences or phrases from a document. These calculations occur with the assistance of pre-created dictionaries which include words annotated with their polarity or semantic orientation. These dictionaries can be manually created or automatically. The automatic generation of these dictionaries uses seed words to expand the list of words [34]. Usually the words that hold the sentimental value are the adjectives [35], which leads to most lexicon-based sentiment analysis applications to focus on them, score them and then calculate an overall score for the whole text.

On the other hand, the machine learning way of creating a sentiment analysis model is similar to any classification problem. Given an input text, the machine learning model decides in which class the input belongs to by comparing it to older instances that it has been trained with. Sentiment analysis can work either with supervised or unsupervised type of learning algorithms. Under the supervised category, texts with their labels are given to an algorithm for training and then, when a new, unlabeled text is inputted, the model outputs the class it predicts. On the unsupervised counterpart of this technique, the machine learning model clusters the data and then provides its’ predictions on new data. It should be noted that there are also semi-supervised approaches [36].

The machine learning approach is considered to be more complicated than the lexicon-based approach, due to the fact that many different algorithms can be used and provide very different results. The Naïve Bayes family of probabilistic algorithms are some of the most popular for sentiment analysis purposes, which can be combined with other parameters and methods like stop-word removal and lemmatization in order to provide accurate results. Support Vector Machines (SVM) [37] as well as Linear and Logistic Regression algorithms [38] are also suitable for these kinds of activities. Chapter 5.4 elaborates further on some of these methods and their implementation in the present project and application, explaining the work and the thought process behind the decision of classifier running the engine of the model used.

As previously described, various factors - linguistic and other- can be involved in the expression of emotion and sentiment in a sentence or document. These factors contribute to the complexity of the procedure and may lead to wrong verdicts, due to the inability of a machine to understand in total the human language, making sentiment analysis a challenging task [39]. Some of the most challenging obstacles to tackle are the following:

1. Negation handling [40]: One of the most difficult and important issues with machine learning sentiment analysis is the handling of negation words. Negation is a very common linguistic construction that affects sentiment, polarity and overall meaning of a sentence or phrase. Negation words reverse the polarity of a sentence and they are hard to track, but need to be taken into consideration.
2. Context: Sometimes, the context is very important. A primary example of the above is the word “not”. When introduced into a sentence, it should reverse the polarity of a sentence since it is a negation word. This does not happen when used to emphasize. For instance, in the sentence “Not only he is a great basketball player, but he is also a great leader figure in the locker room.”, the word “not” adds to the positive meaning of the phrase.
3. Sarcasm: One of the biggest challenges in sentiment analysis is the detection of sarcasm [41]. Sarcastic sentences are really hard to keep track, since when not knowing the context in which a phrase is said in, it is not clear even to humans if the sentence is sarcastic or not. For example, the sentence “Pretty pumped that I woke up at 6am today. Totally how I wanted to spend my Saturday morning!” is said in a negative manner, but a sentiment analysis model would probably classify it as positive. Many researcher teams have tried to improve sarcasm detection, usually with a deep learning approach, achieving satisfying results and showing the potential of future solutions. [42]
4. Ambiguous words or sentences [43]: There are some words that can hold either positive or negative polarity. By having them by themselves sometimes, it is not clear, confusing the sentiment analysis models into wrong predictions. An example is the word “unbelievable”. It can either mean “this is so good I cannot believe it”, or the complete opposite. Sometimes, it does not hold any sentiment at all, being completely neutral. An example of that is the phrase “What happened to Jim is really unbelievable”. It expresses shock and awe, but it is neither positive nor negative. The result of the sentiment that these words hold is based solely on the developer. When using lexicon-based approaches, it takes the value that the creator of the dictionary defines. On the other hand, with machine learning, the model decides on the whether the word holds a positive or negative value depending on the label of the sentence it was in. If it appeared multiple times in a positive sentence, then trying to classify the word “unbelievable” will most probably return a positive class.
5. Abbreviations and acronyms: In the social media age, abbreviations and especially acronyms have been used in a great extent. Due to the character restrictions of Twitter, users have been resorting to acronyms to express themselves. An example of this is the acronym “LOL” which means “Laughing Out Loud”. While some of the acronyms and

abbreviations are unique, there are others that may use the same letters and express different things. Another issue with this category is that the list of acronyms used is extended every day, with new terms appearing, making it difficult for sentiment analysis models to keep up without constant updates.

6. Mistyped words: Typographic errors have always been an issue with natural language processing in general, and sentiment analysis is no different. Spelling correctors have been implemented but are not always flawless, with context playing a major role in the prediction of the word that the user wanted to input. [44]

Many of the obstacles presented above have been researched and been dealt with up to a certain extent [45]. This does not mean that they do not pose a problem, especially to developers and computational linguists new to Sentiment Analysis, who need to do a lot of research to implement solutions for these issues on their own custom models.

4. PROJECT DESCRIPTION AND USE CASES

The present Thesis focuses on the project of creating a Sentiment Analysis application intended for non-experts and a broader user-group. Here, we describe the outline of the system and the constructed platform, the targeted goals and user-groups of the application. We present the textual datasets constituting the backbone of the proposed application along with typical key examples.

4.1 Outline of the System

Sentiment Analysis requires linguistic expertise and can be often be a daunting task for humans and machines alike. It should be taken into account that the majority of end-users of Sentiment Analysis applications are usually neither linguists nor computer scientists and program developers and are usually not very familiar with essential tools and elements of Sentiment Analysis such as Machine Learning. The project and application described and implemented in the present Thesis is intended as a user-friendly web interface platform and middleware for the broad user-group of Sentiment Analysis applications.

The platform developed in the scope of the present Thesis is named, 'How do you feel?', and offers an easy to use, user-friendly user interface where users are provided with the following functionalities:

1. Analyze their own text: The interface provides a textbox that a user can type or paste a text of their choice, or even use an implemented speech-to-text feature to dictate a phrase and then perform on it sentiment analysis. The interface analyzes the text in real time and returns the result to the user.
2. Extract other peoples' opinions from a list of social media networks: While extracting the polarity of an inputted text could provide benefits to a user, Sentiment Analysis is especially remarkable when it can be applied on a wide variety of data while simultaneously, providing statistics about a specific topic. This facilitates the study of the way people think and the way they express themselves. Thus, the second feature of the platform is the possibility to communicate with a variety of websites that people use to express their opinions. Specifically, these websites are: Twitter, the popular micro-blogging social network, Reddit, a collection of forums with an overwhelming number of topics and subjects and, finally, Youtube, the most popular video streaming platform. The exact functional and technical description, as well as usage information of the implementation of data retrieval from these websites will be discussed in Chapter 6.

These functionalities have to be powered by a sentiment analysis mechanism that actually processes the text and outputs the polarity of the documents. A machine learning approach was used, described in detail in Chapter 5.

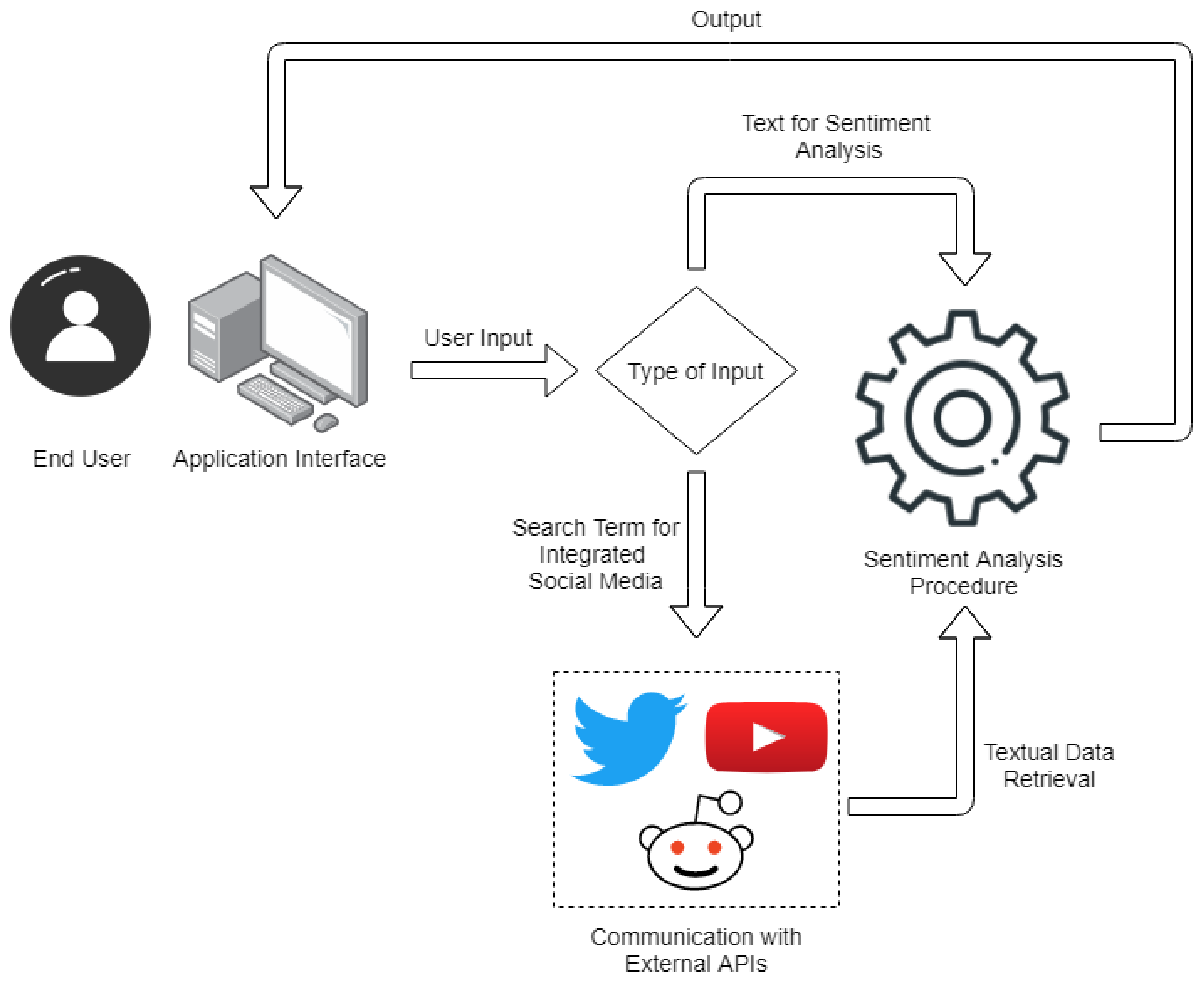

Output
Text for Sentiment Analysis
User Input
Type of Input
End User
Application Interface
Search Term for Integrated Social Media
Sentiment Analysis Procedure
Textual Data Retrieval
Communication with External APIs

Figure 3: High level solution diagram

4.2 Text type and Target User Group

One of the most important questions when wrestling with the idea of what kind of analysis the application should provide (sentimental, subjectivity etc.) is the target audience, the group of users that will use the functionalities of the application. [46] Some examples of target audiences and the way they can use the platform to their advantage are the following:

1. Business Owners / Marketing teams: When a business offers a product or a service and wants to get feedback from the users, Sentiment Analysis can help extract information from huge amounts of data, without the effort of a human. Understanding and seeing how people react to a product can help by providing effective decision making and promote new ideas that may come from random users.
2. Polling / Politicians: A great example of getting sentimental data about polls are election periods. Few months before the actual voting of the public, all the parties can extract information about what the users feel about the political campaign and enrich or, maybe, take a turn on things that do not go well with the flow.
3. Planning purchases / General Public: While the project is about general use and the model is not based on item reviews, (the dataset used to train the model will be discussed in Section 4.4) people can gather valuable information about products or get a general idea of what people think

about specific future purchases they plan to make. Most of the time, it will be helpful and knowing the feelings of people that own the product/use the service.

4. Training- Academic purposes: Course instructors intending to explain to students with examples how Machine Learning works and what “you can do” with this technology, could find great use of this platform, especially due to the simple and easy to navigate interface it offers. Users wishing to experiment with Sentiment Analysis and see “how things work”.

The above-described user groups, as well as any interested parties, are able to use the platform to perform sentiment analysis on text data. While any input can be analyzed, the application is of particular efficiency and performs better when the texts belong to the following categories:

1. Social media quotes: Any opinion expressed on social media networks like Twitter or Facebook is ideal for sentiment analysis. People often resort to their social media accounts to voice their beliefs on various subjects or products and services. Mining the sentiment from these kinds of texts is one of the main functions of the application.
2. Reviews: People criticize products or services, with well-structured text bodies that express their positive and negative impressions. These reviews can be inputted into the application and extract accurate results, especially when it comes to movie reviews.

The above categories are aligned with the datasets used to train the proposed machine learning model and are described in Chapter 5.

4.3 Datasets

A necessary decision preceding the choice and/or design of the classification model is the choice and type of datasets: which datasets are going to be used for the training of the machine learning algorithm, which extends to the type of data the model is designed to analyze.

Since the textual data will come of multiple sources -like the variety of social media the platform offers- and a user will be able to input his own text with no restrictions whatsoever, two datasets were processed and used to train our machine learning models. The contents of the datasets will be described here, as well as the reason they were selected in the scope of our project, concluding with a few examples from each dataset. The datasets used are the Sentiment140, containing textual data extracted from Twitter, and the IMDB dataset, concerning movie reviews extracted by the IMDB website. The implementation of the models will be presented in Chapter 5.

4.3.1 Sentiment140 Dataset

Since the project is revolves mainly around the integration of social media and the extraction of people’s opinions from them, it was mandatory for the machine learning model in the heart of the application to be able to analyze these textual inputs. Twitter has been in the target of Sentiment Analysis research since the earliest days, due to the quite straight-forward way it works and the easy way to extract texts.

Tweets are up to 280 characters long, making them ideal for sentiment analysis and natural language processing experiments. The basic characteristic of tweets is that they are short in length and usually contain a lot of abbreviations and acronyms. Furthermore, mistyped words or repeated characters often are making an appearance in tweets, making the preprocessing of the text necessary. Tweets most of the time contain sentimental information since people usually use Twitter to complain or express their positive emotions and opinions in general, it is not used as a chatting service, making the data originating from it very valuable for Sentiment Analysis.

Gathering data and labeling it manually is a time consuming yet crucial process, since the data need to be balanced and well chosen. For our model, we used the dataset from Sentiment140 which is a popular Sentiment Analysis project [47]. The dataset contains 1.6 million tweets targeted to be completely balanced between sentiments and opinions, with 800 thousand being positive and 800 thousand being negative. There is no “neutral” label and this will be addressed in Chapter 5. The dataset contains a large amount of information such as dates, users and is tagged with a label “0” if the sentiment of the tweet is negative and “4” if it is positive. To visualize the words of the dataset, we created three wordclouds. These wordclouds present the most used words of the whole dataset, the most used words of the positive tagged tweets and, finally, the most used words of the negative tagged tweets.

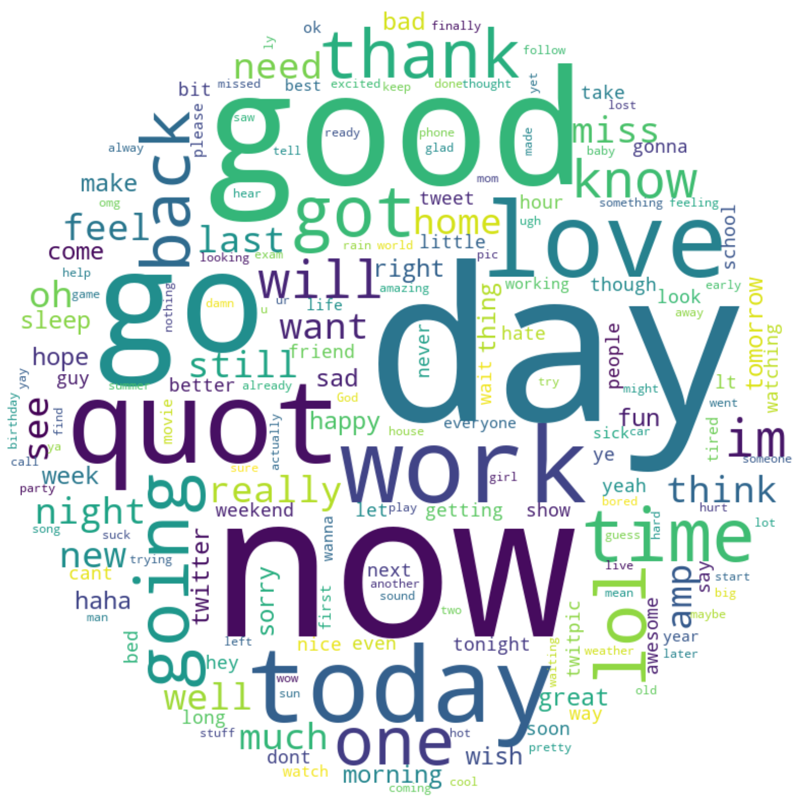


Figure 4: Wordcloud of the whole Sentiment140 dataset

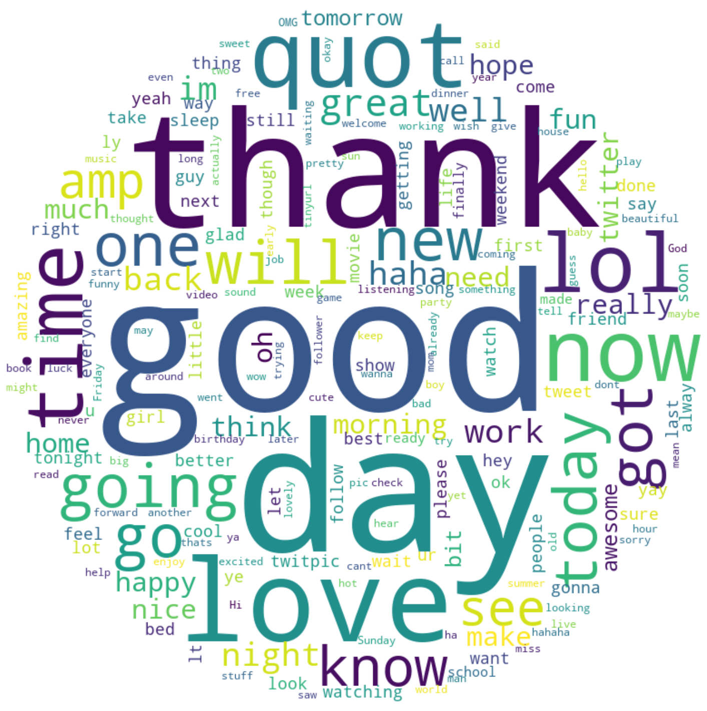


Figure 5: Wordcloud of the positive words of theSentiment140 dataset

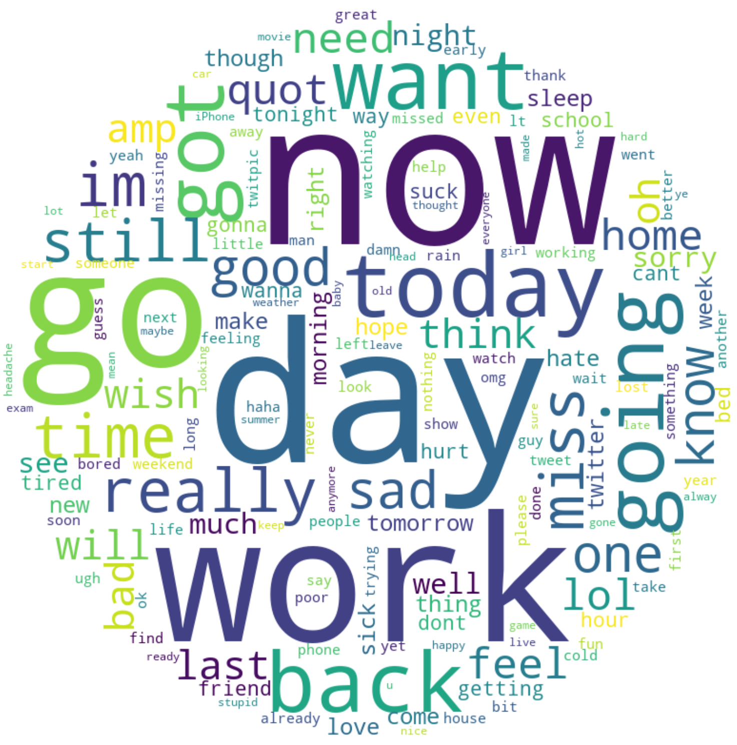


Figure 6: Wordcloud of the negative words of theSentiment140 dataset

It is evident that many of the words in the negative sentiment should not be there. Words like “good”, “better” and “love” express positive feelings and this is an issue that the classification model will have to tackle: Many sentences will contain both positive and negative words and the classification based on solely the words themselves can be a difficult task.

Some selected entries of the dataset are presented and discussed here, in order to present and describe the main components of the textual data:

1. “I feel bad for doing it”: Showing disappointment in carrying out a specific action, the polarity of this sentence is clearly negative. The word “bad” is in almost every context and instance, negative. All of the other words in the sentence can be considered neutral, they do not bear any sentiment value. They are either articles or nouns that in most cases do not have any sentimental significance [23].
2. “Need a hug”: This individual example is a case where the expression does not contain any words that express sadness, anger, or any negative emotion, however, a sad, melancholy feeling even gloom is expressed [23].
3. “Give someone you love a hug today!”: Contradicting the previous example, which also talked about a “hug” in a negative way, this sentence is clearly positive. The exclamation mark, as well as the verb “love”, charge the sentence with a warm sentiment [23] .
4. “My feet hurt… totally worth it though”: Expressing a negative feeling in the first part of the phrase, negating the polarity to hold a positive value with the second. This is considered a common case, especially with negation words like “not” and “no” [23].
5. “I love weekends!”: A clear, positive sentence. The word “love” is positive and the existence of it in the sentence, while the other words are neutral and do not hold any sentimental value, is enough to make this sentence positive [23].
6. “Excited for tomorrow night, going to see the Hannah Montana movie”: A positive sentence due to the “excited” keyword in the beginning of the sentence. The remaining words are neutral, stating the plans of the author, which do not hold any sentimental value [23].
7. “Ugh....92 degrees tomorrow”: Dissecting this example, all of the words do not hold any value. The second part indicates the weather conditions of the following day, which is a neutral statement. The first word of the example, the word “Ugh”, while is not technically a word but an exclamation, expresses discomfort and displeasure in general, which transforms this whole phrase into a negative example [23].
8. “Life is cool. But not for Me”: The opposite situation of the example #4. While the first part of the sentence hints that the polarity will be positive, with the word “cool” indicating happiness, the following segment reverses the total sentiment of the phrase due to the negating word “not” [23].

Since the dataset consists of 1.6 million entries, the above-presented sample is a very limited fraction of the complete set of texts used to train the model. However, there a remarkable accurate prediction of sentiments expressed in the types of texts processed. Tweets contain user’s thoughts and are sometimes randomly structured and these examples provide a precise view of the data.

4.3.2 IMDB Dataset

The first dataset contained short texts, and the second selection of training data is completely different and opposite from the above-described first one. The complementary dataset contains movie reviews extracted from the well-known online movie repository, IMDB. The dataset, similar to its sentiment140 counterpart, was assembled in scope of another project and was released on the web for use in other sentiment analysis applications [48]. It contains 50 thousand movie reviews, with half of the dataset being positive and the other half being negative.

The selection of this dataset to be included in this Thesis was based on the fact that since the platform offers a “Free-text” section, where the user can input any text of choice, some users may select to input long, well-structured texts to quickly sentimentally analyze them. While the first dataset would still be accurate to an extent, the model trained with similar data would provide better results. For this reason, to offer a better user experience, both datasets were included.

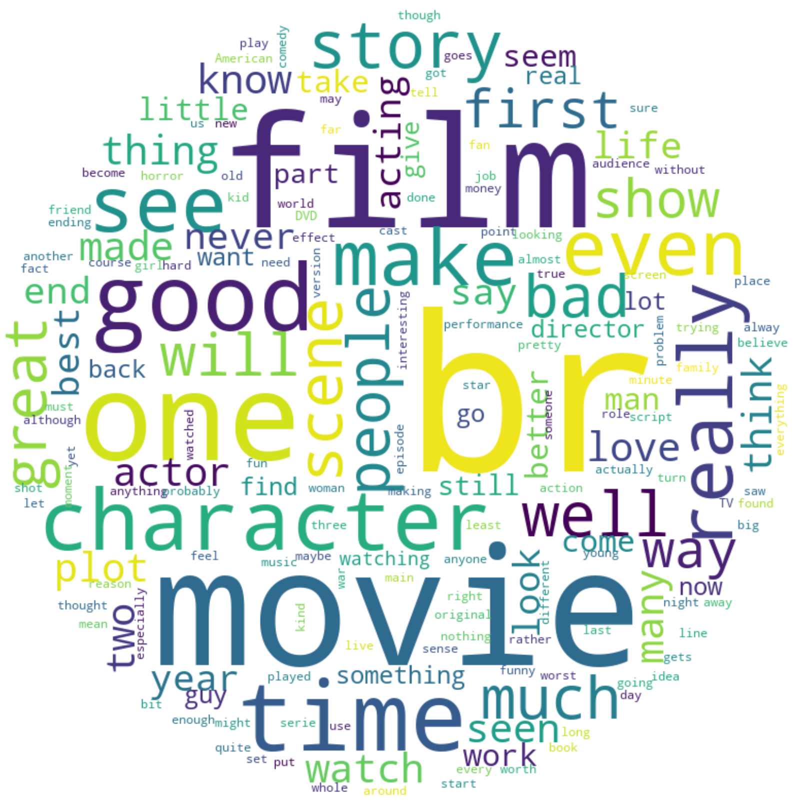


Figure 7: Wordcloud of the whole IMDB dataset

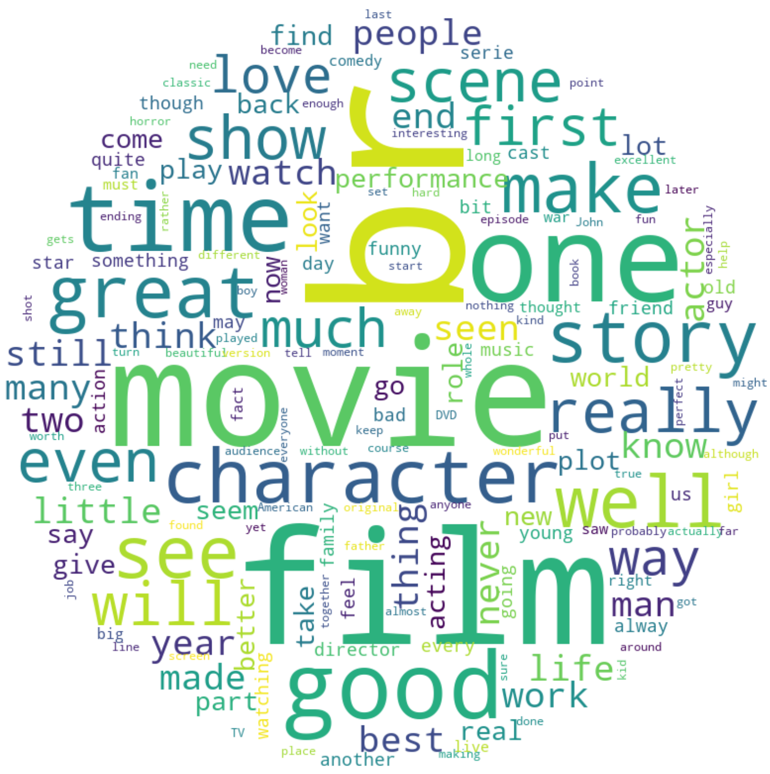


Figure 8: Wordcloud of the positive words of the IMDB dataset

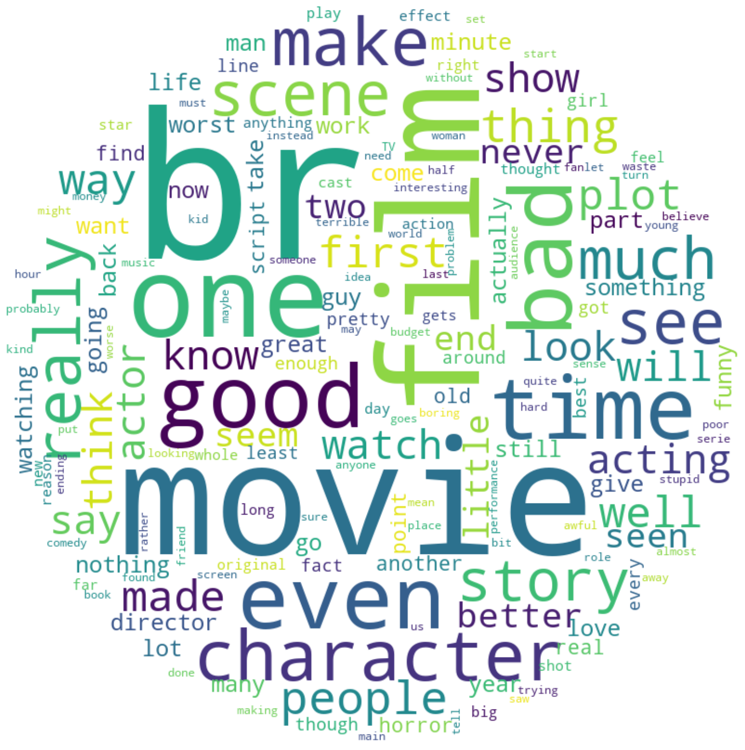


Figure 9: Wordcloud of the negative words of the IMDB dataset

Similarly to the sentiment140 dataset, it is noticeable that many words of the positive labeled data appear in the negative labeled data as well. It is noted that words like “movie”, “film” and “character” are popular since the dataset contains cinematography terms.

Since the majority of reviews are lengthy, the shorter reviews have been selected and will be discussed, similarly to section 4.3.1:

1. “It's terrific when a funny movie doesn't make you smile. What a pity!! This film is very boring and so long. It's simply painful. The story is staggering without goal and no fun. You feel better when it's finished”: The first example of the IMDB dataset is clearly negative. There are some words that express positive feelings like “smile”, “feel better” and “fun”, which are negated either by words like “not”, or are used in a negative context, like the last sentence of the example.
2. “A rating of "1" does not begin to express how dull, depressing and relentlessly bad this movie is”: Undoubtedly, almost all of the core words of this phrase are negative.
3. “I have seen this film at least 100 times and I am still excited by it; the acting is perfect and the romance between Joe and Jean keeps me on the edge of my seat, plus I still think Bryan Brown is the tops. Brilliant Film.”: A positive instance of the IMDB dataset. Every sentence in this particular example holds a positive sentiment.
4. “This film takes you on one family's impossible journey, and makes you feel every step of their odyssey. Beautifully acted and photographed, heartbreakingly real. Its last line, with its wistful hope, is one of the more powerful in memory”: The way this example is dissimilar to the previous is that the opinion of the review’s author is expressed in the middle sentence, with the other two giving context and information on the movie.
5. “I had before a feeling of mislike for all Russian films. But, after seeing this film I haven´t. This is a unique masterpiece made by the best director ever lived in the USSR. He knows the art of film making, and can use it very well.”: The final example presented, shows an instance where the final sentiment is positive but the first part is negative, expressing how the author believed he would dislike the movie. Then the positive polarity is seen, with compliments for the director, swapping the whole polarity of the instance.

A major difference between the two datasets is that the IMDB contains entries that are considerably larger and better structured, with a lot of words, which makes the decision of whether an entry is positive or negative more complex than the sentiment140 dataset. This issue is aimed to be resolved by algorithms helping the prediction model accurately classify the majority of inputted texts into the correct category. The algorithms are described in Chapter 5.

4.4 Greek Data

A major question when discussing the project is whether the application should process and predict the polarity and emotion of other languages, except from English. Greek, of course, was an important consideration for the machine learning model. For example, for Greek sentiment analysis there are not many available sentiment analysis datasets in the Greek language. Several issues arise with the processing of Greek data, in addition to typical issues concerning syntax and grammar, which will not be discussed here. The machine translation of Greek data and its subsequent processing by Sentiment Analysis applications trained for English data, although a relatively quick and easy solution, comes with its own set of shortcomings. Many well-trained machine translation models, such as the Google Translate web application, often output wrong translations because of the many peculiarities of the Greek vocabulary. Although Google Translate uses a neural machine translation model, incorrect translations continue not to be rare. which can provide misleading results. Erroneous translations prove that even the best trained and optimized neural networks encounter problems with less highly resourced languages like Greek. Typical examples are the following:

1. Translating words with scientific context sometimes is a challenging task. Specific terminology is often mistranslated. An example is the term “Εισόσωμα”. It is a term that defines a specific genus of insects, which should be translated to the word “Isosoma”, but is wrongly translated to the work “Insole”.
2. Commonly abbreviated words, with “ό,τι” being a prime example. The word “ότι” has similar meaning to the adverb “that”, but the comma signifies that the word is the abbreviated version of the word “οτιδήποτε”, which means “anything”, with Google Translate translating it to “that”.
3. Words with different meaning according to the context they are used in are also challenging for the machine translation procedure. The word “κατεβάζω” can be translated to “download” correctly, but can also mean “bringing you something downstairs”. The sentence “Σου κατεβάζω τα κλειδιά τώρα” translates to “I’m bringing you the keys downstairs now”, with the machine translation of the sentence being “I am downloading your keys now”, using the wrong verb.

There is plenty of research on Google Translate’s erroneous results, especially when attempting to translate texts to English, from a less widely used language like Greek or Indonesian, explaining the source of the errors and presenting more results and examples [49].

5. PREDICTION MODEL

The present Chapter focuses on the presentation and analysis of the main modules constituting our machine learning model, the decision process behind the algorithm selection, while demonstrating the basic features with examples and statistical analysis. First, the programming language as well as the libraries and modules used will be presented, continuing to the description of the preprocessing function to which every text to be sentimentally analyzed will be submitted. Then, feature extraction methods will be discussed, followed by a brief description of the classification algorithms used for the machine learning model's implementation. Finally, we will describe the implementation procedure of our models and conclude the Chapter with evaluation of the models created on sample data.

5.1 Technical Information

Typical starting points of the development process are the selection of the programming language, as well as the tools to assist in attaining the target product. For this specific component of the project, the machine learning prediction model, the selected programming language for its implementation is Python. Python is a high-level programming language, considered to be approachable and easy to understand and maneuver even to novice programmers not familiar with software development. For that particular reason, Python is used by not only by software engineers but also by statisticians, data analysts, computational linguists and experts of other science fields to use the power of programming to their advantage. Furthermore, a wide variety of external libraries and modules serves a multitude of purposes, such as statistical algorithms and machine learning models, natural language processing related functionalities among others. For the prediction model implemented for the scope of this project, the main libraries that were used were:

1. Scikit-learn (or sklearn): Scikit [50] contains a remarkably large amount of machine learning algorithms and solutions, making it one of the best machine learning libraries. Classification, regression and clustering solutions are offered and is designed to interoperate with the Python numerical and scientific libraries NumPy [51] and SciPy [52]. Scikit's prebuilt functions are used for the machine learning model and feature extraction.
2. NLTK: Standing for Natural Language Toolkit [53], NLTK is a Python module that provides and API for various NLP tasks such as part-of-speech tagging and translation. It offers helpful functions such as sentence tokenizers that are used constantly in natural language processing tasks. It also offers a pre-trained sentiment analysis model out of the box, for users that do not want to create their own custom model. While this could have been a solution for this project, it was decided to go with the custom model to explore the customization options.

5.2 Data Preprocessing

As with any NLP task, data preprocessing is a crucial and important step to improve the classification model, by removing any anomalies and noise, normalizing the textual data. Noise can refer to words that have zero value for the sentence, punctuation and symbols, URLs and, in general, text that provides no polarity weight. Selecting the correct and appropriate preprocessing methods, machine learning models can predict correctly with greater accuracy and score [54] [55]. The model will be trained with the data described in Chapter 4 and, as previously described, the sentiment140 dataset contains 1.6 million tweets, which may contain a lot of unnecessary information. The preprocessing techniques used on the train datasets will also be used for the cleaning of the text that the user inputs, before its classification. Since there is no barrier to what a user might type or what will the integrated services retrieve from the available resources like Twitter, preprocessing will play a major role to ensure that the data going into the model are “clean”. The preprocessing methods used are:

1. Switching all the letters to lower-case: This helps avoid issues where words with different capital letters are treated as different words by the model during training, which may lead to inaccurate results.
2. Removal of punctuation: While sometimes punctuations like exclamation marks might be helpful to a sentiment analysis model, working as a multiplier to the polarity of the sentence, sometimes are also used ironically. To avoid confusion, it was decided not to work with punctuation at all and use a regular expression to remove every instance of punctuation.
3. Removal of symbols: Texts extracted from social media usually contain a lot of symbols. Hashtags (#) are used to tag a post on a social media network to be searchable with the word following the hash. For example, #SentimentAnalysis, when clicked, would prompt a search for the ‘SentimentAnalysis’ term on the website. While hashtags are removed, the words are kept because many times they hold sentimental value. In general, symbols and emojis are removed.
4. Removal of usernames: In contradiction to the words following the hashtag symbol, words following the “@” symbol refer to usernames. For example, ‘@cgs23’ refers to the user ‘cgs23’, which does not provide anything to our analysis. With that being said, every word following the @ symbol, as well as the symbol itself, are removed completely from the document.
5. Removal of URLs: URLs do not offer any sentimental value to a text, so they are completely removed from each instance.
6. Normalization of the words. Text normalization is the process of transforming text into a single canonical form that it might not have had before. Usually two techniques are used for normalization: Stemming and/or Lemmatization [56].
    a. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form, meaning that the suffixes of the words are removed.
    b. Lemmatization in Computational Linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.

   One of these techniques will be chosen for the development of the present application.

5.3 Feature Extraction

With datasets “cleaned” and preprocessed, an essential step before the selection of the classification algorithm for training, the training data needs to be transformed into a more manageable way for processing. This procedure, referred to as feature extraction [57], transforms the documents into vectors, making it easier for a classifier to be trained. These vectors are derived from textual data, in order to reflect various linguistic properties of the text, with features being defined as symbolic characteristics of a text. Vectorizing the data also offers more benefits such as search optimization, helping with the speed of the classification [58]. Feature vectors can be created using several methods. Two methods that were into consideration when implementing the prediction model will be presented here: the Bag-of-Words (BOW) and the Term Frequency – Inverse Document Frequency (TF-IDF) techniques, which are the most popular for NLP tasks in general.

5.3.1 Bag of Words

When it comes to the easy understanding of representations of textual features with vectors, the most common approach is the Bag of Words model [59]. In this model, a text is represented as a bag of the words that it consists of. Specifically, the term “bag” refers to a multiset of words, disregarding grammar and word order and storing the number of occurrences of the individual words (or tokens) of the document. Then, a numerical feature vector is created representing the text. There are many parameters and options when using a Bag of Words model, like the inclusion of n-grams. In the fields of Computational Linguistics and probabilities, an n-gram is a contiguous sequence of n items from a given sample of text or speech. Using Latin numerical prefixes, an n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram" etc. An example of bigrams (N=2) is the following, given the sentence “This is the best movie ever”, the bigrams are:

- “This is”
- “Is the”
- “The best”
- “Best movie”
- “Movie ever”

N-grams serve as tokens and their existence in the bag of words model is identical to single words. Also, the bag of words works as a binary model, checking the existence of a single word instead of the number of occurrences. This is mentioned because it serves some classification algorithms that only look for the presence of a token in a sentence, like the Bernoulli Naïve Bayes that will be discussed in section 5.4.1. An example of the Bag of Words feature extraction is the following:

Considering the two sentences

- This is a nice bag (1)
- This is very fun (2)

With the bag of words technique, these two sentences create the following vocabulary:

1. This
2. is
3. a
4. nice
5. bag
6. very
7. fun

The feature vector representation of the sentence (1) is [1,1,1,1,1,0,0] and the representation of the sentence (2) is [1,1,0,0,0,1,1]. The number 1 indicates the existence of the word, while the number 0 shows the absence of the specific token.

Although the "Bag of Words" (BoW) approach is considered fairly easy to understand and implement, some drawbacks exist. [59] Since each word adds a dimension to the vector, the methods lead to a high dimension feature vector, due to the large size of the total vocabulary of words that exist in the dataset, creating delays in the training process. In addition, when not parametrized properly, it takes into consideration words that do not offer any sentimental value to the document.

For the implementation of a BoW model, the sklearn Python library offers an implementation called Count Vectorizer. Count Vectorizer provides all the necessary functionality to convert a collection of text documents to a matrix of token counts, with several customization options, like the inclusion of n-grams and the removal of stop words, which are words like “and”, “the”, “him”. These words are presumed to be uninformative in representing the content of a text, and they may be removed to avoid them being construed as signal for prediction. The parameterization of the Count Vectorizer will be discussed in the evaluation section of this Chapter, supported by results and statistics.

5.3.2 Term Frequency – Inverse Document Frequency

While the Bag of Words approach, focuses on the existence of a word in a document and the times it appears, the Term Frequency – Inverse Document Frequency approach (or TF-IDF for short), is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [60]. TF-IDF is a common term weighting scheme in Information Retrieval, that has also found good use in document classification. This importance of each word, also referred to as the score of the word, is calculated by multiplying the two terms that make TF-IDF, which are Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency shows how frequently an expression is found in a document, which indicates how significant the expression is to the document. There are many types of term frequencies, like Boolean or term frequency adjusted for document length. The one that is being used in our case is the logarithmic term frequency, defined as follows:

$$tf(t,d) = \log\left(1 + freq(t,d)\right)$$

Inverse Document Frequency, on the other hand, represents the amount of information that a word provides to the document, for example, if it is rare or common across all documents, with rarer words having a higher IDF score. The IDF score is computed as the logarithm of the number of the documents in the corpus, divided by the number of documents where the specific term in question appears. The mathematical definition of IDF is the following:

$$idf(t,D) = log\left(\frac{N}{count(d \epsilon D : t \epsilon d)}\right)$$

As previously described, the TF-IDF is calculated by multiplying the TF with the IDF:

$$tfidf(t,d,D) = tf(t,D) \cdot idf(t,D)$$

The importance of each word increases proportionally to the number of times it appears in the document but is also decreased by the frequency of the word in the total corpus of documents, meaning that the higher the score, the more relevant that word is in that particular document.

Similarly to the the Count Vectorizer implementation provided by sklearn, the library offers also a TFIDF Vectorizer. In our case, since we experimented with the Count Vectorizer, there is a way to transform the resulting feature vectors of the Count

Vectorizer into TF-IDF vectors with the assistance of a TF-IDF transformer, which transforms a count matrix (provided by an instance of a Count Vectorizer) to a normalized TF-IDF matrix.

5.4 Classification Algorithms

When features are extracted and converted in a way that can be used to train a machine learning algorithm, the classification model is chosen. While there are many suitable options of a classification model for Sentiment Analysis and we can treat it as a typical classification problem, there are a few factors to be taken into consideration [61].

It was previously described that we wanted to classify our data as neutral, though there is no neutral class in our datasets. While we could have used datasets that contained the neutral class, we decided to treat this issue in a different way, using probabilistic algorithms. With probabilistic algorithms, when the probabilities of the predicted classes are below a certain threshold, the text is labelled as neutral. This automatically disqualifies a wide range of classification models like Decision Trees and Support Vector Machines due to the fact that they are not probabilistic and only give one specific result.

We also have to take into consideration the fact that our application is served online, so the predictions have to be swift. Especially when working with the Twitter, Reddit or Youtube data, where the sentimental analysis of hundreds of documents will be processed in a single request, we have to serve the results to the client as quickly as possible. This automatically excludes classification models such as K-Nearest Neighbors due to long processing times.

With the aforementioned in mind, the final selections for the prediction models were the Naïve Bayes classifier family, specifically the Bernoulli and Multinomial variations, as well as the Logistic Regression algorithm [62].

5.4.1 Naïve Bayes

All the classifiers that belong to the Naïve Bayes family are based on the Bayes Theorem, which describes the probability of an event happening, based on prior knowledge of conditions that might be related to the event. The mathematical equation that states this theorem is the following

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

with A, B being events, P(A) and P(B) being the probabilities of observing the events A and B respectively and, finally, P(A|B) being the probability of event A occurring, given B is true. The assumption made here is that the predictors/features are independent. A presence of one particular feature does not affect the other. Hence, it is called “naïve”.

There are multiple classifiers that are based on the Bayes theorem. Some commonly used Naive Bayes classifiers include the Gaussian Naive Bayes classifier, the Multinomial Naive Bayes classifier and the Bernoulli Naive Bayes classifier. Since the Gaussian variation is particularly efficient when used on continuous data, the ones taken into consideration for the implementation of this model are the Multinomial and Bernoulli classification algorithms, which both perform especially well in document classification and generally in natural language processing tasks like in our case, sentiment analysis.

Multinomial Naïve Bayes [63] is considered to be one of the most appropriate and efficient algorithms in document classification. The classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts [64], which means that it estimates the conditional probability of a particular word given a class as the relative frequency of the given term, belonging to a class. The multinomial variation's main approach is that it takes into account the number of occurrences of a term in the training data in each particular class, including multiple occurrences. As far as the Bernoulli Naïve Bayes classifier is concerned, like his Multinomial sibling, the Multinomial Naive Bayes classifier, this classifier is suitable for discrete data. The difference is that while the Multinomial Naïve Bayes classifier works with occurrence counts, the Bernoulli Naïve Bayes classifier is designed for binary features. It checks for the existence of values instead of assigning it a different weight for the amount of times it appears in a positive/negative sentence.

An evaluation of both of these classifiers will be presented, with the only difference being that when using the Multinomial Naïve Bayes algorithm, the feature extraction technique utilized will be the TF-IDF method. This has no extra value for the Bernoulli Naïve Bayes classifier, since it only concerns the existence or absence of a token in a document. For the Bernoulli algorithm, a Bag of Words model will be used. Finally, it should be taken into consideration that the Multinomial Naïve Bayes algorithms work better with small datasets.

5.4.2 Logistic Regression

In most of the cases, when implementing a document classification model, the Naïve Bayes algorithms are usually the most common approach. For binary classification, Logistic Regression, another algorithm, is borrowed from the field of Statistics. Logistic Regression is considered to provide high-quality results and high accuracy. Named after the function that runs at its core, Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function. The logistic function, also known as sigmoid function due to having a characteristic 'S' shaped curve, has the following mathematical equation:

$$f(x) = \frac{L}{1+e^{-k(x-x0)}} \text{ , where}$$

$x0$ = the x value of the sigmoid's midpoint,
L = the maximum value of the curve and
k = the logistic growth rate or steepness of the curve.

This function is used in the Logistic Regression classification algorithm to predict the probability of an event based on the values of one or more explanatory variables.

Logistic Regression [65] functions better with bigger datasets and provides probabilities for the outcomes, which assists the model developed in this project as mentioned above.

5.5 Training procedure

The above-described preprocessing and feature extraction techniques, as well as the classification algorithms are followed by the testing and evaluation of the combination of methods. There are many parameters that affect the success rate of the model and they do not only consist of the data that trained the algorithm. Since the number of all combinations of parameters is remarkably large, the decisions regarding the parameters will be taken in the following order:

1. Firstly, the data will be preprocessed. stemming or lemmatization will be selected based on the better results. All of the other preprocessing techniques presented in Chapter 5.2 will be used in all times.
2. After the selection of stemming or lemmatization, the feature extraction function will be tuned. Count Vectorizer will be used, using the TFIDF transformer to convert the Bag of Words results into TF-IDF instances.

3. Finally, one of the three classifiers will be chosen and tuned to yield the best results possible.

Considering all of the above, when experimenting with stemming or lemmatization, the base, untuned versions of Count Vectorizer, TF-IDF transformer and Multinomial Naïve Bayes will be used, to have the same base of evaluation. When tuning the Count Vectorizer, the same untuned classification method will be used.

All of the metrics and statistics provided are based on the same train/test data. When training a model with different data, the results will vary. With that in mind, the data is being mixed in the same exact way every time the model is trained and the train/test data is split by 85%/15%. When the decision about which classification model is final, then the classifier will be trained with 100% of the data.

5.5.1 Normalization

The text normalization techniques, stemming and lemmatization, as described in a previous Chapter, are very helpful and assist in better prediction outcomes. For the stemming, an implementation of the Porter Stemming algorithm [66] was used, that is provided by the NLTK Python module. The same module provides the Word Net Lemmatizer, used for lemmatization. A very helpful technique that also helps lemmatization is Part-of-Speech (POS) tagging. POS tagging is the process of marking up a word in a text (corpus) as corresponding to a particular Part of Speech based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of Part-of-Speech tagging is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc. POS Tagging does not improve the results of the stemming technique so it is only utilized during lemmatization.

For the sentiment140 dataset, the normalization techniques do not present a remarkable improvement; this is because of the structure of the data. Since tweets sometimes tend to be incorrect from a syntax and grammar standpoint, the normalization techniques, while they do work, the improvements are little.

Table 1: Normalization results (sentiment140)

#	Normalization	Accuracy
1	None	74,97%
2	Stemming	75,12%
3	Lemmatization (with POS Tagging)	75,52%

As shown in Table 1, the lemmatization technique offers the best result, by a slight margin, the classifier offered around .40% better accuracy when trained with data that were preprocessed with lemmatization.

The same techniques were applied onto the IMDB dataset. Due to its more structured data, while the differences were not considerable, as shown in Table 2, there was a margin of around 1%, which is 60% more than the improvement we received on the previous dataset. It is notable that even with no further tuning in the prediction models, the results are already satisfying, with 75% and almost 87% accuracy for each classification model. These results will be subjected to further improvement.

Table 2: Normalization results (IMDB)

#	Normalization	Accuracy
1	None	85.77%
2	Stemming	86.28%
3	Lemmatization (W/ POS Tagging	86,77%

5.5.2 Feature Extraction

As a next step, the next component in need of tuning is the Count Vectorizer. The vectorizer offers a wide variety of parameters that can be fine-tuned to offer better results. Some of these parameters are:

1. The number of most frequently used features: Results vary when the vectors include all of the corpus of words, or the most used N words. As shown in results, better accuracy is offered when the whole bunch of words is used.
2. Whether or not to ignore stop words: Though the term "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all Natural Language Processing tools, and, indeed, not all tools even use such a list. The stop word list used is the pre-built English list of the scikit library.
3. Tokenizer: The function that splits the sentence into tokens. Tokens are words and punctuation symbols. A basic word tokenizer is used and also the TreeBank Word Tokenizer, which operates in a slightly different way. The results are almost identical for both tokenizers.
4. N-Grams: Experimenting with multiple numbers of N, the results clearly favor N-grams with N>1.
5. Ignore words that appear in less than K documents: This is a very important parameter because it excludes tokens (or N-grams) that appear a less than a specific amount of times, meaning they are probably mistyped or terminology and should not be considered when making a decision. When experimented with unigrams, K=5 was producing satisfying results. When raising the parameter above K=5, accuracy fell. When N-grams with N>1 came into play, the optimal value of K was around 20-50.

Table 3: Count Vectorizer tuning results (sentiment140)

#	N most frequently used words	Ignore words that appear in less than N documents	Tokenizer	N-Grams	Stop words	Accuracy
1	All (Not Specified)	1	None	(1,1)	No	75,52%
2	All (Not Specified)	2	None	(1,1)	No	77,31%
3	All (Not Specified)	5	None	(1,1)	No	77,411%
4	All (Not Specified)	10	None	(1,1)	No	77.4%
5	5000	2	None	(1,1)	No	76.84%
6	10000	2	None	(1,1)	No	77.2
7	15000	2	None	(1,1)	No	77,36%

8	All (Not Specified)	5	Treebank Word Tokenizer	(1,1)	No	77.43%
9	All (Not Specified)	5	Word_tokenize	(1,1)	No	77.43%
10	All (Not Specified)	5	Treebank Word Tokenizer	(1,1)	Yes	76.023%
11	All (Not Specified)	5	Word_tokenize	(1,1)	Yes	76.023%
12	All (Not Specified)	5	Word_tokenize	(1,2)	Yes	77,2%
13	All (Not Specified)	5	Word_tokenize	(1,2)	No	80,37%
14	All (Not Specified)	5	Treebank Word Tokenizer	(1,2)	No	80,37%
15	All (Not Specified)	10	Treebank Word Tokenizer	(1,2)	No	80.22%
16	All (Not Specified)	20	Treebank Word Tokenizer	(1,2)	No	80%
17	All (Not Specified)	5	Word_tokenize	(1,3)	No	80,37%
18	All (Not Specified)	20	Word_tokenize	(1,3)	No	80.23%
19	All (Not Specified)	20	Word_tokenize	(1,4)	No	80.21%
20	All (Not Specified)	50	Word_tokenize	(1,4)	No	80.23%
21	All (Not Specified)	5	Word_tokenize	(1,4)	No	80,76%

Table 4: Count Vectorizer tuning results (IMDB)

#	**N most frequently used words**	**Ignore words that appear in less than N documents**	**Tokenizer**	**N-Grams**	**Stop words**	**Accuracy**
1	All (Not Specified)	1	None	(1,1)	No	86,77%
2	All (Not Specified)	2	None	(1,1)	No	86,76%
3	All (Not Specified)	5	None	(1,1)	No	86,61%
4	All (Not Specified)	10	None	(1,1)	No	86,16%
5	5000	2	None	(1,1)	No	85,45%
6	10000	2	None	(1,1)	No	85,68%
7	15000	2	None	(1,1)	No	86.01%
8	All (Not Specified)	5	Treebank Word Tokenizer	(1,1)	No	86,61%
9	All (Not Specified)	5	Word_tokenize	(1,1)	No	86,61%
10	All (Not Specified)	5	Treebank Word Tokenizer	(1,1)	Yes	86,626%
11	All (Not Specified)	5	Word_tokenize	(1,1)	Yes	86,626%
12	All (Not Specified)	5	Word_tokenize	(1,2)	Yes	88,44%
13	All (Not Specified)	5	Word_tokenize	(1,2)	No	89,026%
14	All (Not Specified)	5	Treebank Word Tokenizer	(1,2)	No	89,026%
15	All (Not Specified)	10	Treebank Word Tokenizer	(1,2)	No	88,6666%
16	All (Not Specified)	20	Treebank Word Tokenizer	(1,2)	No	88,36%
17	All (Not Specified)	5	Word_tokenize	(1,3)	No	89,48%
18	All (Not Specified)	20	Word_tokenize	(1,3)	No	88,89%
19	All (Not Specified)	20	Word_tokenize	(1,4)	No	88,973%
20	All (Not Specified)	50	Word_tokenize	(1,4)	No	88,186%
21	All (Not Specified)	5	Word_tokenize	(1,4)	No	89,6933%

In both scenarios, both models yield the best results when tuned with the parameters shown in row #21. The accuracy has increased by a significant amount in both cases, 5% in the sentiment140 model and slightly less than 3% in the IMDB model. The verdict from these evaluations and tests is that fine-tuning the feature extraction function can provide vast improvements. Another observation is that both our models have had the best accuracy with the same parameters.

5.5.3 Classification Algorithm

For our final experiment, using the two decisions from our previous evaluations, we will assess the classification algorithms. As we mentioned, we will take a look at the two classifiers that belong in the Naïve Bayes family, the Multinomial classifier and the Bernoulli classifier, as well as the Logistic Regression classifier. This is our most important decision, so we will not depend only on the accuracy metric to make a decision. Accuracy is the easiest metric to rely upon and understand while evaluating a data model. Classification accuracy shows us the number of correct predictions made divided by the total number of predictions, multiplied by 100 to convert into a percentage. The major concern with accuracy is that sometimes it can be misleading, showing inaccurate success rate percentages. Since accuracy sometimes is not always... accurate, and we want a deeper look into the results, we will also calculate precision, recall and F1-score.

Precision attempts to answer the following question: What proportion of positive identifications was actually correct? And recall attempts to answer the reverse question: What proportion of actual positives was identified correctly? Finally, the F-Measure provides a single score that balances both the concerns of precision and recall in one number. Therefore, using the F-Measure we will have taken into consideration both recall and precision.

For both the datasets in question, various combinations of parameters of the Multinomial Naïve Bayes classifier, Logistic Regression classifier and Bernoulli Naïve Bayes classifier were tested.

Table 5: Multinomial Naive Bayes tuning (sentiment140)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	alpha=1.0,fit_prior=true,class_prior=none	80.2308	80.6222	79.6436	80.1299
2	alpha=1.0,fit_prior=false,class_prior=none	80.2316	80.6189	79.6511	80.1321
3	alpha=1e-1,fit_prior=true,class_prior=none	80.2441	80.6822	79.5820	80.12841
4	alpha=1e-1,fit_prior=false,class_prior=none	80.2458	80.6788	79.5920	80.1317
5	alpha=1e-2,fit_prior=true,class_prior=none	80.2462	80.6991	79.5604	80.1257
6	alpha=1e-2,fit_prior=false,class_prior=none	80.2466	80.6957	79.5670	80.1274
7	alpha=0,fit_prior=false,class_prior=none	80.2437	80.7002	79.5521	80.1220
8	alpha=2,fit_prior=false,class_prior=none	80.2308	80.6248	79.6395	80.1291

Table 6: Bernoulli Naive Bayes tuning (sentiment140)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	alpha=1.0, fit_prior=True	78.7312	76.9040	82.1878	79.4581
2	alpha=1.0, fit_prior=False	78.73	76.9022	82.1878	79.4572
3	alpha=2.0, fit_prior=False	78.6566	76.7540	82.2735	79.4180

4	alpha=0, fit_prior=False	78.8179	77.0736	82.099	79.5072
5	alpha=1e-1, fit_prior=False	78.8037	77.0506	82.1045	79.4973
6	alpha=1e-2, fit_prior=False	78.8187	77.0739	82.1012	79.5082

Table 7: Logistic Regression tuning (sentiment140)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	default	82.4638	81.8635	83.4507	82.64945
2	max_iter=200	82.4779	81.8800	83.4607	82.6628
3	max_iter = 300	82.4733	81.8794	83.4499	82.6572
4	max_iter = 200, solver='newton-cg'	82.4746	81.8829	83.4474	82.6578
5	max_iter = 200, solver='saga'	82.4742	81.8823	83.4474	82.6574
6	penalty='elasticnet', max_iter = 200, solver='saga', l1_ratio=0.5	82.5470	81.8852	83.6292	82.7482

Table 8: Multinomial Naive Bayes tuning (IMDB)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	alpha=1.0,fit_prior=true,class_prior=none	89.693	89.3018	90.0910	89.694
2	alpha=1.0,fit_prior=false,class_prior=none	89.706	89.325	90.0910	89.70666
3	alpha=1e-1,fit_prior=true,class_prior=none	90.373	89.527	91.349	90.429
4	alpha=1e-1,fit_prior=False,class_prior=None	90.373	89.527	91.349	90.429
5	alpha=1e-2,fit_prior=True,class_prior=None	90.3866	89.6134	91.2694	90.4338
6	alpha=1e-2,fit_prior=False,class_prior=None	90.3866	89.6134	91.2694	90.4338
7	alpha=0,fit_prior=False,class_prior=None	89.28	89.402	89.01981	89.2109
8	alpha=2,fit_prior=False,class_prior=None	89.28	89.402	89.01981	89.2109

Table 9: Bernoulli Naive Bayes tuning (IMDB)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	alpha=1.0, fit_prior=True	86.746	86.428	86.029	86.25
2	alpha=1.0, fit_prior=False	86.862	86.474	86.029	86.29
3	alpha=2.0, fit_prior=False	86.046	86.028	85.929	86.03
4	alpha=0, fit_prior=False	86.046	86.028	85.929	86.03
5	alpha=1e-1, fit_prior=False	86.946	86.824	86.528	86.71
6	alpha=1e-2, fit_prior=False	87.037	86.532	86.439	86.682

Table 10: Logistic Regression tuning (IMDB)

#	Classifier Parameters	Accuracy	Precision	Recall	F Measure
1	default	90.0933	89.07	91.2961	90.173
2	max_iter=200	90.0933	89.07	91.2961	90.173
3	max_iter = 300	90.093	89.077	91.29	90.1732

4	max_iter = 200, solver='newton-cg'	90.056	89.02	91.04	89.48
5	max_iter = 200, solver='saga'	90.106	89.100	91.29	90.1851
6	penalty='elasticnet', max_iter = 200, solver='saga', l1_ratio=0.5	88.9866	87.963	90.2249	89.0798

In both cases, the instance of Bernoulli Naïve Bayes algorithm, while performed well, came up short in comparison to the other two classification algorithms, showing us that frequency of word appearance does matter and treating the data as binary does not always help, especially in the case of words that hold sentimental value as the features of the classification problem. The Multinomial member classifier of the Bayes family does perform well in all cases, especially out of the box, without much tuning, showing once again why it is considered one of the best classification algorithms when dealing with textual data in a machine learning problem, rendering the best results when dealing with a smaller amount of data (IMDB model). The Logistic Regression algorithm proved to be better when trained with larger datasets, which means it is used in the sentiment140 model. An observation made is that, while tuning the classifiers does not offer great improvements on any of the metrics in question, it is the algorithm that actually makes the bigger difference.

5.5.4 Combined Model

For the sake of comparisons and experimental reasons, a model created by the combination of the two datasets was also created. Since the normalization techniques and the count vectorizer parameters were the same in both the first two models, we decided to only experiment the classifiers during training. An instance with the Logistic Regression model, tuned as was in the sentiment140 prediction model and another one with the Multinomial Naïve Bayes of the IMDB model were created. As depicted in Table 11, the sentiment140 – Logistic Regression model yielded better results than its IMDB – Multinomial Naïve Bayes counterpart. This was expected; as stated in the description of the classification algorithms, the Logistic Regression classifier deals with larger datasets in a better way than the Multinomial Naïve Bayes classifier, scoring better in all four of the metrics.

Table 11: Combined dataset classifier scores

#	Classifier	Accuracy	Precision	Recall	F-Measure
1	sentiment140	82.779	82.3014	83.709	82.99
2	IMDB	80.7131	81.38	79.7962	80.58

5.6 Evaluation

Our three models are trained and ready to be used for Sentiment Analysis. To test and evaluate these models, we will present some examples of texts and the label that was outputted by the models, for comments and comparisons between the three.

Since our models only predict the classes that have been trained with, thus making “positive” and “negative” the only possible results, after it was decided to classify as “neutral” the textual inputs that the models do not have more than 75% confidence in any label. For instance, if a text is classified as positive with a 0.6 probability by our model, the results shown to the user will be neutral.

Table 12: Sentiment analysis examples

#	Input	Sentiment140	IMDB	Combination
1	This is an amazing project.	Positive (91,1%)	Positive (97,2%)	Positive (95.8%)
2	I would like you to inform me about the next flight dates to Belfast, please.	Neutral (60,6% positive)	Neutral (73,8% negative)	Neutral (59% positive)
3	This has been an awful year.	Negative (88,2%)	Negative (80,6%)	Negative (94,5%)
4	Donald Trump is the president of the United States of America.	Neutral (73% positive)	Neutral (74.8% positive)	Neutral (69% positive)
5	This is not good at all	Negative (97,6%)	Negative (93,4%)	Negative (97,8%)
6	I loved the movie even more than the book!	Positive (81,4%)	Positive (95,6%)	Positive (78,7%)

To comment on the results of the analysis of the examples presented in Table 12:

1. Example #1 is a simple, positive sentence. The models easily predict the correct label, with high positive percentages.
2. Example #2 is a request which does not hold any polarity. While the results are correct, it can be derived that the IMDB model almost classifies the sentence as negative. This is probably due to movie reviews in the dataset expressing negative comments about a movie containing some of these terms.
3. Example #3 is a negative comment about the current year, correctly classified by all the models.
4. Example #4 states a fact. Facts should be neutral and do not hold any sentiment and while the models predict correctly the result, all of them almost classify the sentence as positive. This is due to the dataset containing the terms of this sentence with a positive context, which may lead to wrong predictions.
5. Example #5 shows a negation example. This example is predicted correct due to the fact that n-grams have been included in the models. The bigram “not good” is negative, which helps with the correct classification of the sentence. Counting only on unigrams, this sentence would have been classified as positive by all the models due to the existence of the word “good” in it.
6. Finally, Example #6 shows the difference between the classification of inputs that are suited for a specific model. The input refers to a movie, which makes the prediction from the IMDB dataset much more confident than the other two models.

Although all of the three models provide satisfying predictions, they still may produce errors, especially in occasions when a neutral fact is expressed and there is no clear polarity in the sentence. Since the classification of the final result is probabilistic, the threshold of 75% might need to be adjusted in future situations. Nevertheless, the probabilistic approach seems to function correctly to a great extent.

5.7 Summary

In this Chapter we presented the thought process, development information and research behind the machine learning models concerning our application. First, we presented the frameworks, modules and programming language used for the implementation of the prediction model, with Python and sk-learn being at the core of the application. We then described the preprocessing function and the preprocessing techniques it utilizes to cleanse and transform a textual input into a shape that contains only valuable information to the sentiment analysis machine learning model, discarding words and text features that do not hold any sentimental value.

Furthermore, feature extraction techniques were discussed; Bag of Words and Term Frequency – Inverse Document Frequency were the ones selected for the present project and application. In addition, some classification algorithms which are popular in Sentiment Analysis were discussed and presented. These are the Multinomial and the Bernoulli Naïve Bayes classifiers, as well as the Logistic Regression algorithm.

Finally, all of the above was used in experiments with the two datasets presented in Chapter 4 to create the final three (3) prediction models of ‘How do you feel?’. All of the data was preprocessed with the same function, the TF-IDF method was selected for feature extraction and the classification algorithms selected for the models were: Multinomial Naïve Bayes for the IMDB dataset and Logistic Regression for the remaining two models (Sentiment140 and the combination of the two). The final models were then evaluated using random English phrases, yielding satisfying results with accuracy percentages of around 85%.

6. COMMUNICATION – DATA ACCESS LAYER

In order for the three implemented machine learning predictions models to be offered to the end-user, there has to be a data access layer, where the user input is given to the classification model to bypass preprocessing, vectorization and have its class predicted by the selected model.

While presenting the data to the end-user is a procedure every developer has to be very elegant about, the communication between the application and the database (and in our case, the classification model and the external networks) has to be well structured, safe and quick. This Chapter will discuss the communication implementation, with information on how the application manages to retrieve data from the social media networks and analyze them sentimentally in real time.

6.1 Technical Information

Since our machine learning models were built using Python, the decision to create the communication System of our application using a Python framework was an easy one. The framework chosen to be the backbone of our application is the Django framework [67], enriched with the Django Rest Framework module [68]. Django is a popular full stack web development library, suitable for the front and back end of web applications. The Representational State Transfer (REST) software architecture that the Django Rest framework offers provides interoperability between computer systems on the internet, thus making the Application Programming Interface (API) of our application completely independent from the interface, meaning that it can also be used by other web applications or API clients like Postman.

The main object of this API is to receive textual inputs and use the classification models described in the previous Chapter to serve the prediction results to the user. The biggest issue to be dealt with is that the application needs to be swift, since the user does not want have long wait times for the results to appear on the screen. The duration of a machine learning model training is long, sometimes exceeding even the 24-hour mark, depending on the algorithm and datasets. With execution time being one of our biggest concerns, a solution to the problem needed to be found. Since the models are stored in the run time memory, they are deleted when the Python script that trained them is terminated and trained again when the script is fired up. To avoid these training delays, Pickle was used.

Pickle is used for serializing and de-serializing Python object structures, also called “marshalling” or “flattening”. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network. In other words, when pickling a Python object, it is retrieved from the Random-Access Memory (RAM) and saved into the disc, granting the ability to be loaded later with a lot less time complexity. Pickle is a Python module served with the main distribution of the programming language. Therefore, when the API is executed, it takes around 20 seconds to load the models and have them ready for predictions, in comparison to the hours or even days it would take to retrain the models on the fly.

Since the Application Programming Interface will be served over the internet, security plays a major role in the healthy operation and functionality of the services. There are two major approaches when it comes to securing APIs. The first is to provide a registration form for anyone that wishes to use the web services, providing expiring tokens for each session. While this is considered to be a great way to deal with security, the application that will consume the API will be accessible to anyone without the need for registration, which automatically moves us to the second method to deal with the security issue, which is API Keys. Using the Django Rest framework API Key module

[69], we are able to create an API Key that gives access to the functionalities of the REST API and have it hidden into the application's configuration files. This way, our application will be the only one that can communicate with the API, avoiding any efforts for malicious usage. The API Key is inserted into the authentication header of the requests directed to the API, allowing the owners of the correct key to communicate with the implemented data retrieving and Sentiment Analysis functions.

6.2 External API Communication

As most Sentiment Analysis applications, the proposed 'How do you feel?' application is observed to perform well when used on a lot of data, to extract verdicts and conclusions about how people feel, retrieve their opinions and use this information to one's advantage. A way to achieve this functionality is to retrieve data from popular social networks, where people express themselves. For the retrieval of the data, while crawling and scraping techniques would have worked, it would have been difficult to clean the results and costly in terms of time. To confront this issue, communication with external APIs was chosen to be implemented. Since some of the most widely used social networks such as Facebook and Instagram do not offer their data accessing APIs to the public, the choices narrowed down to Twitter, Reddit and Youtube. All three of these options can contain opinions on products, services or even politics, which is beneficial when using a Sentiment Analysis application.

To access all of the APIs of the web pages mentioned above, developer accounts had to be created to get API keys. All of these API Keys (which are mandatory for the authentication phase during the communication with the endpoints) are stored into three separate configuration files, one for each Application Programming Interface provider. The configuration files serve as an extra layer of security in order to avoid hard coding credentials and authorization tokens into the code that requests the resources from the endpoints and in order to be manageable in case there is a need to change them.

6.2.1 Twitter

Twitter, being one of the most researched and studied social network when it comes to Sentiment Analysis, was one of the basic pillars on which the application 'How do you feel?' was built on. To retrieve tweets and the information of the authors, for convenience, Tweepy [70] was used. Tweepy is a Python module that communicates with ease with the Twitter API, fetching data in an easy and feasible way. It converts the response JSON in a Python object form, making it easy to manage and obtain the valuable parts.

Twitter API's responses hold plenty of information [71]. One request for a tweet returns a JSON response that contains information about the tweet (such as the time of the post, language and geographic information), statistics (such as number of retweets) and favorites, but it also contains details about the user that posted the tweet. These details feature if the user is validated or not, user name and location that the user states in his profile, number of followers and accounts following and many more. The fact that this amount of information is returned in a single request is convenient, needing less requests even when requesting a huge number of tweets to be returned.

Tweepy assists with the type of tweets that the application fetches. It filters out retweets, since we do not want to include duplicates and also helps with the language of the texts, since our application works with English data.

6.2.2 Reddit

Similarly to Tweepy, for the retrieval of data from Reddit, the Python Reddit API Wrapper library (PRAW) [72] is utilized. PRAW offers multiple functionalities such as search per user, subbredit or even with submission and random search terms in the whole reddit website. After authenticating with API keys, provided by the configuration files mentioned earlier, data is conveniently acquired in the same manner as Tweepy; Python objects that can be iterated are created.

One major issue with the Reddit Application Programming Interface is that the responses, on the contrary to Twitter's implementation, contain much less information. For example, the request for a specific comment contains the comment's details like time and the actual text, as well as a unique ID for the user that posted it. To retrieve the user's information, a separate request has to be made. Also, we are considering only the top comments in our data retrieval function. This is because the replies are additional requests and, furthermore, when replying to a top-level comment, sometimes the content of the reply does not hold sentimental value. Taking all the above into consideration, for a large amount of data, multiple requests have to be made, which leads to delays due to a bottleneck; Reddit allows maximum 60 requests per 1 minute, resulting in our application responding slower than usual when using the Reddit functionalities.

6.2.3 Youtube

As a member of the Google API ecosystem, retrieving data from Youtube's API [73] is effortless. Google offers a well-structured solution, making the communication very easy, with the Google API Python Client library [74] connecting and serving the data the users request quickly and easily. This Application Programming Interface offers a modular approach; the request can be modified to ask about more information in one request. The parameter that allows this functionality is named "part" and it can be modified to include statistics about videos, comments, user and channel information, as well as playlists and many more features.

While Google does restrict the API requests to around 8000 per day, these are more than enough to satisfy the needs of our application. Since the threshold is daily and not by the minute, it does not affect the speeds of our application.

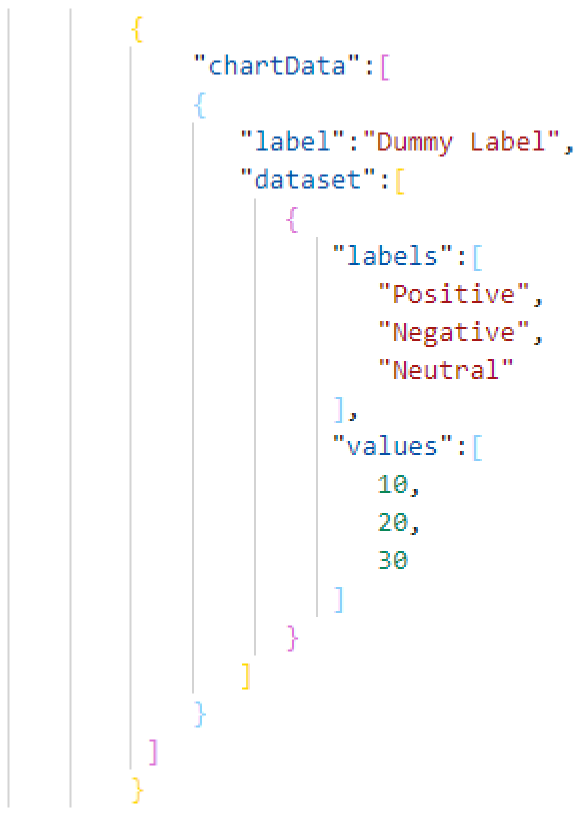
6.3 API Endpoints

With the variety of functionalities the application offers, the API had to be dissected into six different endpoints. Each endpoint serves a different purpose, sharing the same structure when it comes to request headers and HTTP methods.

First of all, all of the following API methods support only the POST HTTP method. Since the user is submitting a form from the front end to the back end of the application, the POST method the only way to go. The body of the request differs per method, with the only common feature being that the 'Content-Type' header is 'application/json', since we are formatting the user input to a JSON object. Moreover, one of the most important request headers is the 'Authorization'. We mentioned before that we use an API Key to authenticate the users of the Application Programming Interface. The syntax of the 'Authorization' header is 'Api-Key XXX', where XXX is the encoded key.

As far as the responses are concerned, they are JSON objects, consisting of two nested objects. These nested objects are:

- The chartData object, which contains the information which will be displayed as charts in the front end of our application. The data is structured in a unique way to be compatible with the Chart.JS library which is in charge of rendering the charts and figures on the user interface. For example, when requesting the sentiment analysis of 10 tweets, a JSON object is created to hold the counts of the positive, negative and neutral tweets, as well as other information that will be displayed via a chart like labels and the chart title. Each chartData object holds four different arrays, one for each chart that is rendered on the front end. Figure 7 shows an example of the data needed for one chart. Since charts are plotted and displayed on all of the three social media related functionalities, these charts hold different, unique information for each social network and the type of information visualized for each service by each individual plot will be discussed in greater detail in the presentation the user interface in Chapter 7.



```
{
    "chartData":[
    {
        "label":"Dummy Label",
        "dataset":[
            {
                "labels":[
                    "Positive",
                    "Negative",
                    "Neutral"
                ],
                "values":[
                    10,
                    20,
                    30
                ]
            }
        ]
    }
    ]
}
```

Figure 10: chartData example

- The tweetData / redditData / youtubeData objects, which contain a more straight-forward look into the fetched data. These objects will be presented in the respective sections of the present Chapter.

The single case that this specific rule does not apply to is the function that classifies a user inputted text and does not display any charts on the interface, or retrieve data from external sources.

All of the functions follow the same logic and structure, to avoid confusions and keep everything well-ordered and inline. Firstly, after the inputs of the request are validated by the corresponding to the method serializer model, the connection to the external APIs is being established by authenticating using the appropriate keys. Then, the data is fetched into a Python objects, iterating through every row and extracting the information needed. After extracting the text to be sentimentally analyzed, it is being preprocessed by a specific function that applies all the methods and techniques discusses earlier, it gets transformed into a TF-IDF vector and is inserted into the classification model. In the end, the data extracted, as well as the prediction of the model and the original text are being packed into a JSON response, containing the chartData and the twitterData / redditData / youtubeData accordingly.

In case of an erroneous API call, all of the functions return an appropriate, well-structured exception message delivered in a JSON object, with the property “errorMessage”. Additionally, to ensure that the data received in the requests are valid, the back end makes use of the Django serializers, which ensure that the inputs received follow specific rules and restrict the users from having the ability to enter any input, narrowing the faulty operation of the API. Following is the description of the endpoints and the functions behind them, giving a detailed view to the way the API operates. Each of the six endpoints represent a functionality of the application, which are offered to the user via separate forms and web pages.

6.3.1 Free-text Endpoint

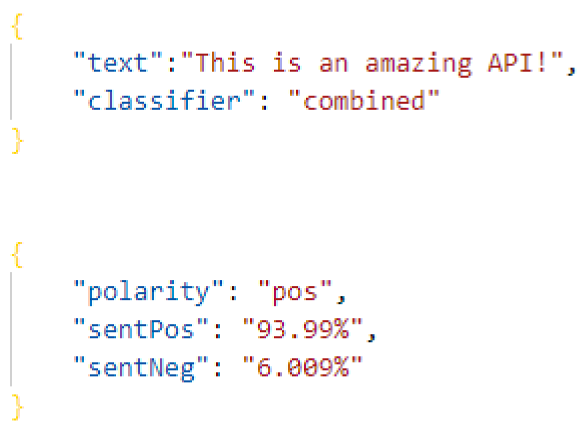
One of the main attractions of the web application interface is the ability to classify any user inputted text. This Application Programming Interface call, which can be reached when communicating with the /api/text/ endpoint, with the appropriate headers described before, submits to the platform a JSON object with the following fields:

- “text”: The input the user wishes to classify. Since it is a ‘free-text’, as described by the application, this field has no validations; the length and content of the text can be anything. Of course, the language should be English.
- “classifier”: The prediction model that the user selects to classify his text with. The input should be one of the following three selections: “imdb” for the IMDB model, “twitter” for the sentiment140 model and “combined” for the model trained with both of the datasets. This field undergoes validation and raises an exception if the user attempts to enter a different selection.

The response of this specific request, as mentioned earlier, does not have the same structure as the other responses due to the absence of charts and the need to collect more data. It returns a JSON object, populated by the fields presented below:

- “polarity”: The polarity of the text after being evaluated by the probability threshold that determines if the text is neutral. The possible values are “pos”, “neu” and “neg” for positive, negative and neutral texts respectively.
- “sentPos” / “sentNeg”: The probability percentages of the positive and negative classes, which in actuality are the true, unfiltered results of the classification process.

The /api/text/ endpoint is the only instance that the order of actions varies; since there is no need to connect to an additional API to retrieve information, the data retrieval and authentication steps are skipped and the text is directly preprocessed and used by the prediction model for the classification.



```
{
    "text":"This is an amazing API!",
    "classifier": "combined"
}
```

```
{
    "polarity": "pos",
    "sentPos": "93.99%",
    "sentNeg": "6.009%"
}
```

Figure 11: /api/text/ example request and response

6.3.2 Twitter Endpoint

The Twitter endpoint, reachable through the /api/twitter/ URI, requires three fields to be filled in order to function properly. These fields are the latter:

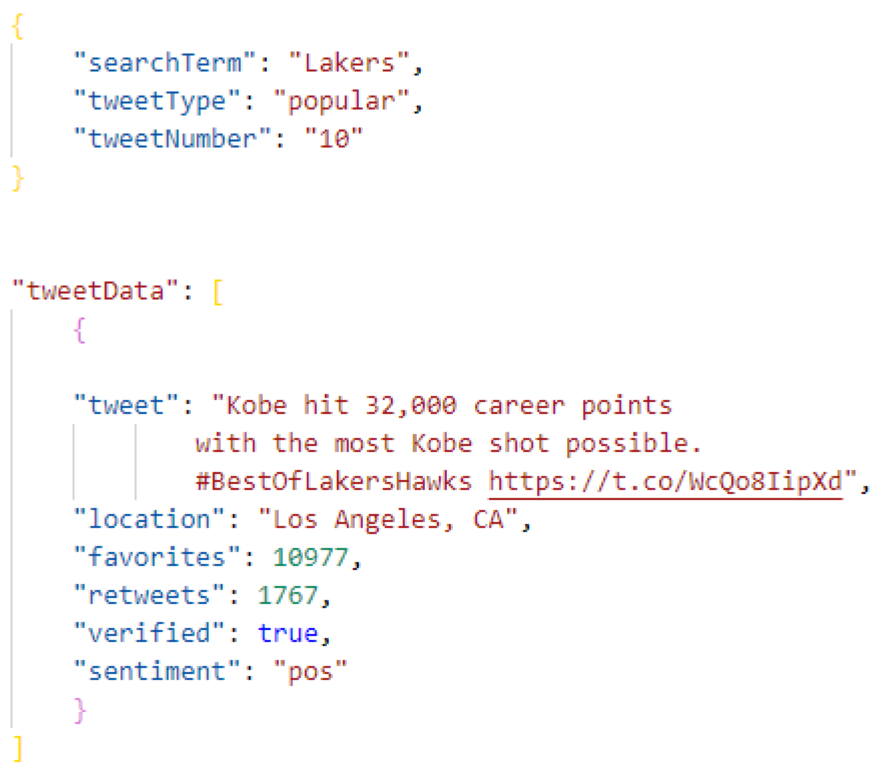
- “searchTerm”: The query to be searched in the Twitter website. It has a max size of 30 characters since it is a keyword and probably inputting more than 30 characters in the search bar does not return any results.
- “tweetType”: Twitter offers three different types of tweets when using the search functionality of its API. These selections are “popular” returning the most popular tweets by means of favorites, retweets and the verified status of the authors being true, “recent”, fetching the newest tweets and “mixed”, which is a combination of the two. These are also the values that the tweetType parameter accepts.
- “tweetNumber”: An integer which indicates the number of tweets to be fetched. The maximum value is 1000.

With the above data, a call is made to the Twitter Application Programming Interface via Tweepy and an object with the results is returned. This object holds a lot of information that is not needed so the function iterates through it to extract the data desired, which is returned via the tweetData object in the response:

- “tweet”: The original tweet, before going through the preprocessing function.
- “location”: The location of the tweet. This location is stated on the user’s profile and is not the live geolocation that the tweet was sent from. While this was considered, when experimenting, it was noticed that a small portion of Twitter users actually reveal their live locations while tweeting which led to lack of data.
- “favourites”: The number that the tweet has been marked as favorite by other users.
- “retweets”: The number of retweets of the specific tweet.
- “verified”: Whether the author is verified on Twitter or not. Verified means that the account holder is a person of public interest and is authentic.

- “sentiment”: The result of the sentiment analysis. This, of course, is not extracted by the Tweepy object but rather added in the end of the procedure by the endpoint’s function.

The data described above, while shape the tweetData object returned in the response, are also divided in separate Python dictionaries and are concatenated to form the chartData object. These dictionaries hold the counts per sentiment of the retweets, favorites, verified users and the locations. The only issue occurring is that many users type same places in a different way, resulting in duplicate locations. For example, “LA, CA” being treated as different than “Los Angeles, CA”. While when just displaying the data in the tweetData object this is not a complication, when counting the different instances of each location, we might end up with same locations with different names.



```
{
    "searchTerm": "Lakers",
    "tweetType": "popular",
    "tweetNumber": "10"
}


"tweetData": [
    {

    "tweet": "Kobe hit 32,000 career points
            with the most Kobe shot possible.
            #BestOfLakersHawks https://t.co/WcQo8IipXd",
    "location": "Los Angeles, CA",
    "favorites": 10977,
    "retweets": 1767,
    "verified": true,
    "sentiment": "pos"
    }
]
```

Figure 12: /api/twitter/ example request and response

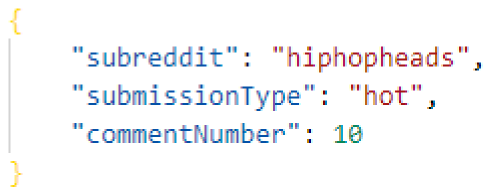
6.3.3 Reddit Endpoints

‘How do you feel?’ takes advantage of three different usages of the Reddit API, offering three distinct endpoints. The URIs of these endpoints are: /api/redditSub/, /api/redditSearch/, /api/redditURL/. Each individual endpoint serves a different functionality, with specific request bodies. The responses returned by these endpoints are identical, which is the reasoning behind the decision to discuss all three of them in the same Chapter.

The /api/redditSub/ endpoint utilizes the “subreddit” function residing in the generic Reddit object of the PRAW library. It searches for a specific subreddit using a variety of parameters, returning an instance of a class named “Subreddit” from which all the requested subreddit’s information can be retrieved, such as title, description and all of the submissions of the subreddit. From the search parameters, three have been selected to be inputted by the interface of ‘How do you feel?’. These parameters, which constitute the body of the /api/redditSub/ request are:

- “subreddit”: The name of the subreddit the user wishes to extract comments for sentiment analysis. The serializer model of this specific term verifies the the input is no longer than 20 characters long.
- “submissionType”: When browsing through the thousands of different subreddits that exist on Reddit, there are multiple ways to fetch the submissions. Some of these sorting methods, which are accepted by the endpoint method are the ‘hot’, which is a ranking algorithm that takes into account several things about the submissions, ‘new’ which fetches the newest submissions, ‘controversial’, meaning submissions with similar counts of upvotes and downvotes. ‘Rising’ fetches the submissions that are gaining upvotes and, finally, ‘top’ which yields the top recent posts.
- “commentNumber”: The number of top-level comments to be fetched from each submission.

When submitting a request to the /api/redditSub/ endpoint, the first 10 submissions of the inputted subreddit, sorted by the selected method described by the “submissionType” parameter are iterated, retrieving a dynamic amount of comments, contained in the “commentNumber” field. From the data retrieved, the comments get preprocessed and sentimentally analyzed, forming the redditData response JSON object.



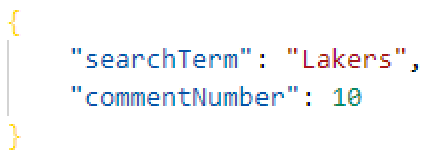
```
{
    "subreddit": "hiphopheads",
    "submissionType": "hot",
    "commentNumber": 10
}
```

Figure 13: /api/redditSub/ example request

Moving forward, the /api/redditSearch/ uses the search function of PRAW, which explores the whole Reddit website via the API for a specific term, returning a list of submissions from various subreddits. The only parameter that the method of our endpoint assigns dynamically to the PRAW search function is the term to be searched. The request body of /api/redditSearch/ consists of:

- “searchTerm”: The keyword that the user wants to search reddit for the top submissions that include it, with a maximum of 30 characters.
- “commentNumber”: As described in the previous call, the number of top-level comments that we want to sentimentally analyze from each yielded submission.

In a similar fashion to the /api/redditSub/, the /api/redditSearch/ returns fetches a number of comments, specified in the “commentNumber” parameter from the first 10 submission results of the search function.



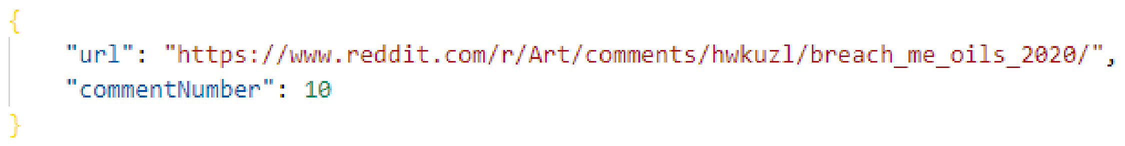
```
{
    "searchTerm": "Lakers",
    "commentNumber": 10
}
```

Figure 14: /api/redditSearch/ example request

Lastly, /api/redditURL is the final of the Reddit family endpoints. It receives the following request body:

- “url”: The URL of the specific submission that the user wants to extract and analyze the comments from.
- “commentNumber”: Similar to the requests of the previous Reddit endpoints, it describes the number of comments to be fetched per submission.

The Reddit client offered by PRAW can be used to retrieve the Submission object from a specific URL. After the user requests a particular URL and the number of comments that wants to be analyzed, PRAW searches and retrieves the requested data. If the URL does not exist, the endpoint method handles the exception, informing the user of the error.



```
{
    "url": "https://www.reddit.com/r/Art/comments/hwkuzl/breach_me_oils_2020/",
    "commentNumber": 10
}
```

Figure 15: /api/redditURL/ example request

All of the three endpoints in the Reddit scope return the same response object, which is constructed by concatenating the chartData object and the redditData object. Like in the previous description of chartData, unique dictionaries that hold the counts of the values presented into the redditData, per sentiment, for each comment. The redditData object contains valuable information which can be used to extract verdicts about the textual, analyzed data. The object contains the following fields:

- “text”: The comment text. It is the original, before applying any preprocessing methods and techniques.
- “sentiment”: The polarity of the comment, after being analyzed by the prediction model.
- “score”: The number of upvotes the comment has received, illustrating its popularity.
- “submission”: The name of the submission, valuable especially to the /api/redditSearch/ and /api/redditSub/ endpoints, since when using the /api/redditURL/ method, the title of the submission is known.
- “subreddit”: The subreddit that the submission originates from.
- “replies”: The number of replies that the comment has received.

- “commentKarma”: Comment Karma, similarly to the “score” of a comment, represents the popularity of the comment’s author. It is a global variable that collects the summary of upvotes that the user has received throughout every reddit submission he has participated actively in.
- “linkKarma”: Likewise, link karma depicts the popularity of the links, such as articles, videos etc. that the user has posted onto reddit.
- “gold”: Each user can become a gold user by paying a monthly fee, giving them bonus functionalities and additional features, such as seeing a breakdown of where your karma came from, which comments you've already read, and the ability to load more comments at once. Additionally, it also gets rid of the ads and a private subreddit. The value returned is either true or false.
- “employee”: This parameter states if the author of the comment is a reddit employee or not.

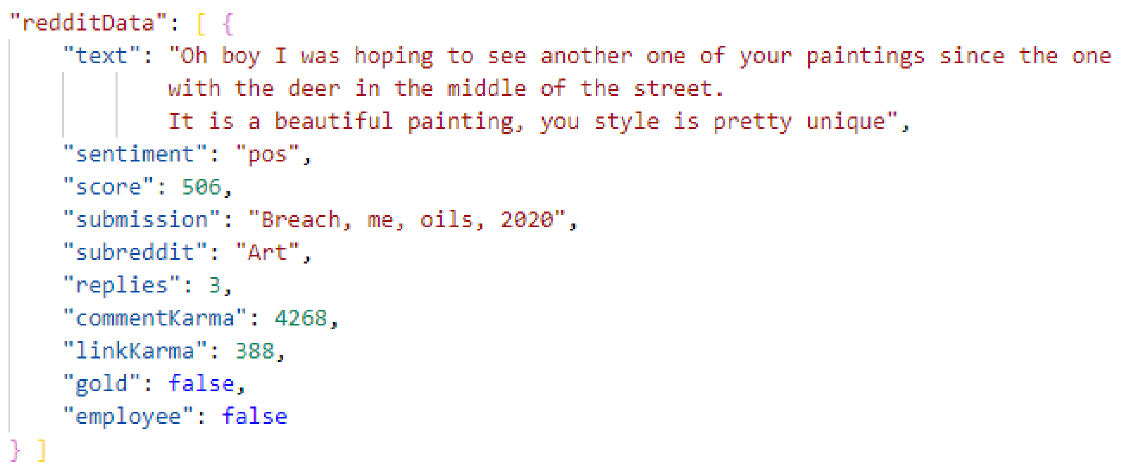

```
"redditData": [ {
    "text": "Oh boy I was hoping to see another one of your paintings since the one
            with the deer in the middle of the street.
            It is a beautiful painting, you style is pretty unique",
    "sentiment": "pos",
    "score": 506,
    "submission": "Breach, me, oils, 2020",
    "subreddit": "Art",
    "replies": 3,
    "commentKarma": 4268,
    "linkKarma": 388,
    "gold": false,
    "employee": false
} ]
```

Figure 16: Reddit endpoints example response

6.3.4 Youtube endpoint

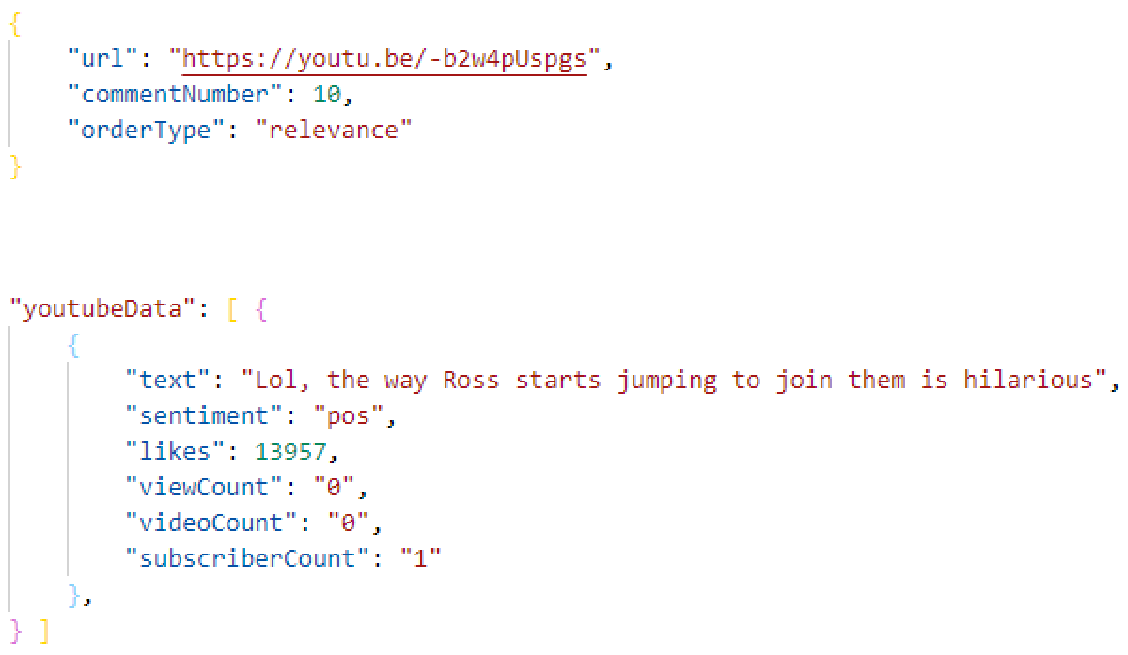
To retrieve information from a Youtube submission, the /api/youtube/ controller has been implemented. As all of the previous endpoints, it user the respective Python wrapper library to communicate with the database of the website, in this case the Google API Python Client library. When requesting the details of a specific video, the response contains multiple arrays of data that assist in minimizing the requests to the Google server. To query for a particular video, the unique identifier of the video is needed, which is specified in the “v” query parameter of the URL. For example, the video “https://www.youtube.com/watch?v=-b2w4pUspgs” is uniquely identified by the parameter “-b2w4pUspgs”. After the insertion of the URL, the user can select the sorting method of the video’s comments, with the two options being “time”, for a descending sorting from newest to oldest, or “relevance”, which displays the more popular comments first. With all the above being said, the parameters needed for the /api/youtube/ call are:

- “url”: The URL of the video the user wants to sentimentally analyze its comments. Youtube offers two types of URLs, the full and the shortened. The function developed to manipulate the URL strings and extract the video ID can handle both of the versions of the video URLs.

- “orderType”: The order of the retrieved comments. The values can be either “time” or “relevance”, which is verified and validated by a serializer.
- “commentNumber”: The number of the top-level comments to examine their polarity.

After requesting the information and receiving the response from the server, the response JSON that will be returned via the /api/youtube/ POST request is structured, consisting, as all the other responses from the chartData and youtubeData arrays. The youtubeData array holds valuable information about the authors of the comments, as well as statistics regarding the comment itself. Since the Youtube users have been merged with their Youtube channels, there are not many statistics regarding the user that we can actually use and study, so we are limited to the channel’s statistics. To be precise, the contents of the object are the following:

- “text”: The original Youtube comment, without any preprocessing techniques applied.
- “sentiment”: The polarity value that was predicted by the machine learning model.
- “likes”: The number of likes the comment received by other Youtube users.
- “viewCount”: The collective number all of the videos the user has uploaded have been viewed.
- “videoCount”: The number of videos the user has uploaded.
- “subscriberCount”: The number of Youtube users that are subscribed to the author’s channel.



```
{
    "url": "https://youtu.be/-b2w4pUspgs",
    "commentNumber": 10,
    "orderType": "relevance"
}
```

```
"youtubeData": [ {
    {
        "text": "Lol, the way Ross starts jumping to join them is hilarious",
        "sentiment": "pos",
        "likes": 13957,
        "viewCount": "0",
        "videoCount": "0",
        "subscriberCount": "1"
    },
} ]
```

Figure 17: /api/youtube/ example request and response

6.4 Summary

The communication layer of the proposed ‘How do you feel?’ application was presented in this Chapter, in particular, the way the data inputted from the user interface is transferred to the back-end of the application and used for sentiment analysis. First, we presented some information about the frameworks and libraries used for the development process. Python, Django and Django Rest Framework are the base of the RESTful API used for communication between the machine learning model and the front end.

The next step was the description of the API endpoints. The API endpoints correspond to each core functionality of the application, them being the ‘Free-text’ sentiment analysis, where the user can input any text of their choice. The Twitter/Reddit/Youtube endpoints are used for data retrieval and sentiment analysis of comments from the respective online social network and displayed some examples of response and request bodies for each endpoint.

6.5 Insights for Greek Data

As discussed in the previous Chapters, the machine learning models implemented and used in ‘How do you feel?’ are trained by and based on datasets containing English textual data. While the implementation of a Greek sentiment analysis model is challenging, if developed as a future update for the designed platform, some modifications will have to be made for it to be integrated correctly into the application.

The language of the inputted text will have to be specified for the appropriate sentiment analysis model to be selected. This feature can be achieved either by (a) prompting the user to input the language of the text he wants to sentimentally analyze, or (b) by developing a function to understand the language of the text received via the ASCII codes of the individual letters. The first option, while easier to implement, adds another input field for the user to provide and increases the error probability due to the human factor. The second option, where the function decides the text’s language, adds a complexity level to the code, but is safer and existing solutions offer very satisfying accuracy percentages while determining the language of a text. [75] [76]

Since developing a Greek sentiment analysis model, trained by a Greek dataset which has been preprocessed appropriately, is a challenging task, another solution that can be suggested is receiving the input in Greek (or any other language) and then translating it in English, to perform sentiment analysis on the translated text. This way, an already implemented model that has been trained by an English dataset can be used.

It should be taken into account that, as previously discussed, machine translation does not always work correctly and accurately, especially in less highly resourced languages such as Greek. Nonetheless, it is an option that can generate satisfying results and can be fairly straight-forward in its implementation, especially if an external translation Application Programming Interface is selected, like the Google Translation API. [77]

7. USER INTERFACE – PRESENTATION LAYER

Collecting and processing textual data from all the available resources described in the previous segments might be complex and frustrating for many individuals with no computer science background, which makes the visualization and presentation of the sentiment analysis procedure one of the most remarkable assets of this application. While the actual analysis of the texts is the most crucial procedure, the way the processed data are presented to the user is also a critical process, helping the users extract the correct conclusions and verdicts on the Sentiment Analysis results through well-designed charts and analytic reports of the analyzed data.

In this Chapter, we will present the frameworks and programming languages used to develop the interface of the proposed 'How do you feel?' project and application. In addition, we will provide some images of the user interface, as well as a brief description of the purpose and functionality of each individual web page.

7.1 Technical Information

While an approach to create a full-stack Python only application was considered, in the end, it was decided to experiment with different technologies and frameworks, thus the front-end of 'How do you feel?' was developed using the ReactJS framework [78]. ReactJS is a JavaScript framework developed by Facebook and is used by major companies and projects around the world.

The main idea behind ReactJS is the reusability of components, making the applications as modular as possible. In the end, the combination of components renders the final view of the webpage. To render a view to a compatible browser, ReactJS uses JSX, an XML/HTML-like syntax that allows JavaScript code and HTML to co-exist in the same file but, unlike past, outdated frameworks, instead of putting JavaScript into HTML, JSX allows us to put HTML into JavaScript. Stepping in this component based direction, the components of the navigation sidebar, the header which contains the logo of the application, the footer and the main styling properties of the application, such as the background color and font families, are single components, rendered in the main component and are not copied over to the other JSX files. This approach is optimal, making the application lightweight and the code reusable, easily understandable and refactorable (improve specific parts of the web application without changing others).

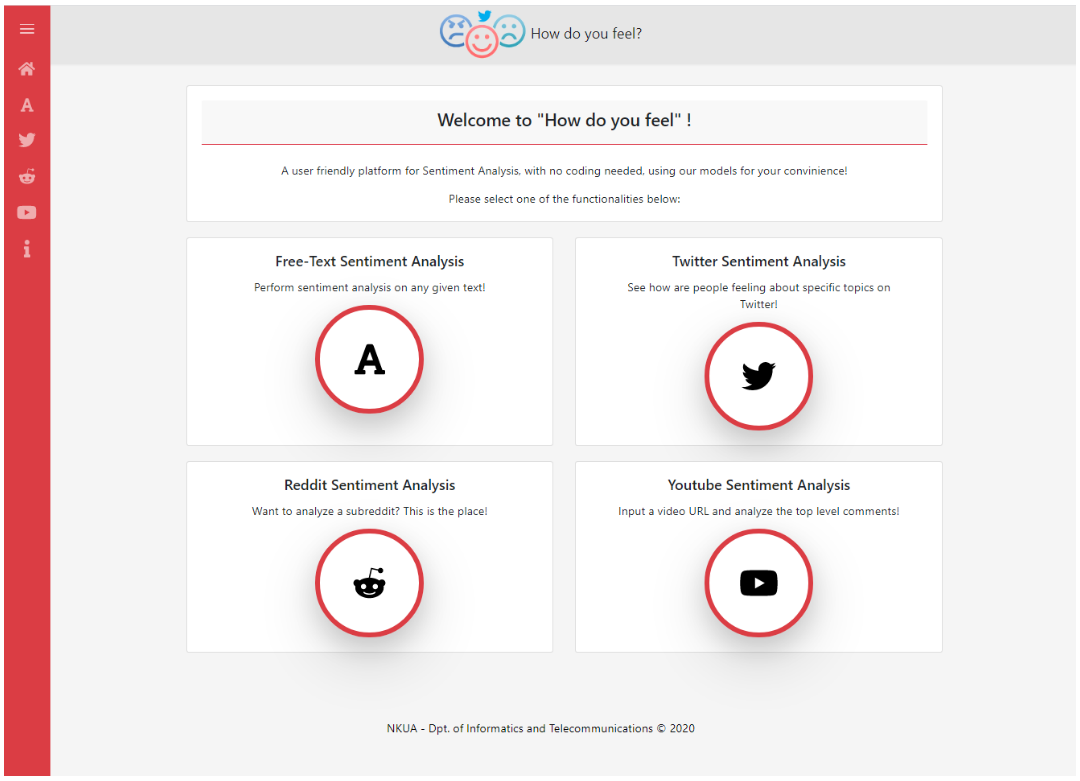
The application offers six (6) different web pages in total, four for each of the functionalities (Free-text, Twitter, Reddit, Youtube sentiment analysis), a homepage as well as an about page. All of these web pages reuse custom components and are styled using Bootstrap, a popular CSS library [79], as well as custom styling sheets for each individual component. For the HTTP call needs of the application, namely the communication with the backend, Axios [80] was utilized.

Axios is an open source HTTP client that returns promises, allowing us to use async methods during the data requests and response receivals. Async requests make the application more responsive and faster, requiring no reloads of the web page to render the results of a request to the browser. This is of particular convenience to the user.

In the following section, the webpages of the platform will be presented, along with a description of the features and functionalities, showing the connection between them and the endpoints described before. Other dependencies required for the application as a whole to function correctly include the React Router, for the navigation and routing between the different web pages of the application, as well as the "fontawesome" icon library for the images and icons displayed on the interface for a more user-friendly experience.

7.2 Homepage

The landing page of the application, accessible through the root URL (“/”), displays buttons which correspond to the four core functionalities of the platform. These buttons perform the same routing action to the respective URI as the navigation sidebar, which is on the left side of the interface. The buttons in the center do not include the “About” page, which is only visible on the sidebar, for aesthetic and symmetry reasons. The icons feature animations achieved via custom CSS, included in the styling sheet file of the homepage component. The sidebar can also be expanded, offering a better view and description of each icon, as seen in Figure 16.


How do you feel?

Welcome to "How do you feel" !

A user friendly platform for Sentiment Analysis, with no coding needed, using our models for your convinience!

Please select one of the functionalities below:

Free-Text Sentiment Analysis

Perform sentiment analysis on any given text!

Twitter Sentiment Analysis

See how are people feeling about specific topics on Twitter!

Reddit Sentiment Analysis

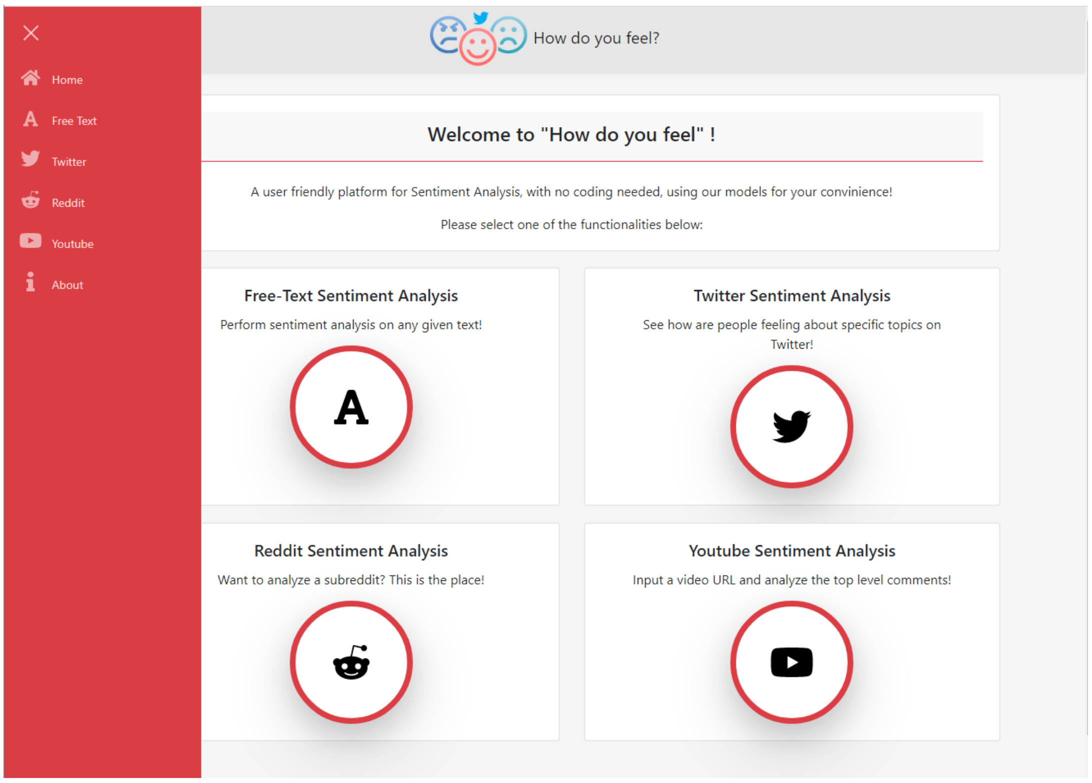
Want to analyze a subreddit? This is the place!

Youtube Sentiment Analysis

Input a video URL and analyze the top level comments!

NKUA - Dpt. of Informatics and Telecommunications © 2020

Figure 18: Homepage layout



Welcome to "How do you feel" !

A user friendly platform for Sentiment Analysis, with no coding needed, using our models for your convinience!

Please select one of the functionalities below:

Free-Text Sentiment Analysis

Perform sentiment analysis on any given text!

Twitter Sentiment Analysis

See how are people feeling about specific topics on Twitter!

Reddit Sentiment Analysis

Want to analyze a subreddit? This is the place!

Youtube Sentiment Analysis

Input a video URL and analyze the top level comments!

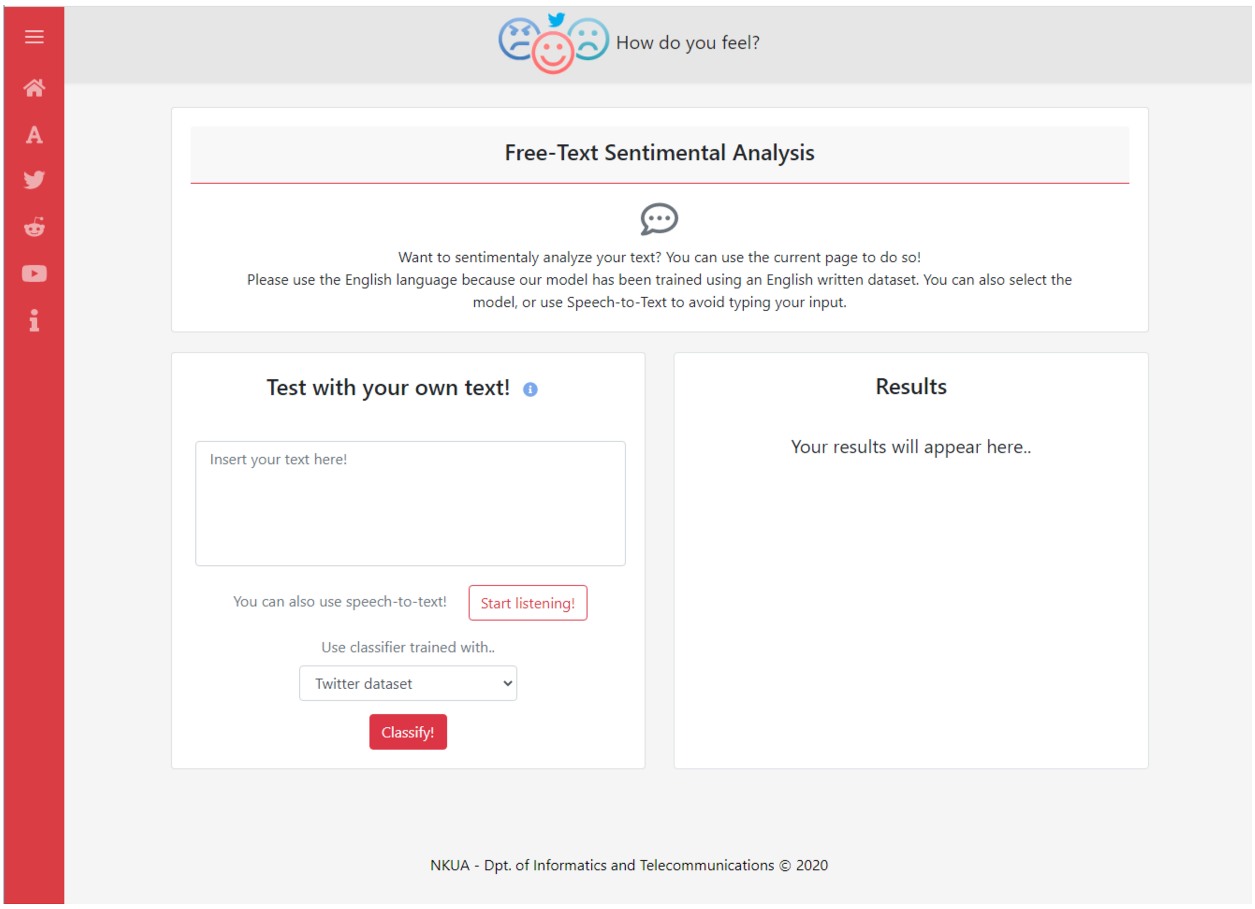
Figure 19: Extended sidebar

7.3 Free-text Sentiment Analysis

The Free-text sentiment analysis part of the platform uses the same square-card layout as the rest of the website. On the left component there is a text area that the user can input the text of his choice for Sentiment Analysis. A dropdown menu is offered for the user to select the model of his choice, from the ones described in Chapter 5. There is also an information button that contains a brief description of the types of texts suitable for the platform, as well as a description of the machine learning models utilized for the analysis process.

The “classify” button calls the /api/text/ endpoint with the text inputted by the user and the machine learning model is selected from the dropdown menu. The button switches to a loading spinner as the browser awaits the response of the API call. When the response is received, the results are shown on the right square, with the boarder changing color according to the result of the Sentiment Analysis process, with red for negative polarity, blue for neutral and green for positive. If there is an error during the communication with the Application Programming Interface, a popup text is shown with the appropriate message that is returned from the API and the HTTP status code that refers to it.

The “classify” button, when clicked, verifies that the textbox is not empty. In the case that it does not contain any characters, it forbids the user from clicking the button and submitting an erroneous request. Finally, there is a speech-to-text functionality, where the user can dictate a text via a microphone and watch it appear live on the text area, giving the user an alternative input method.


How do you feel?

Free-Text Sentimental Analysis

Want to sentimentaly analyze your text? You can use the current page to do so!
Please use the English language because our model has been trained using an English written dataset. You can also select the model, or use Speech-to-Text to avoid typing your input.

Test with your own text!

Insert your text here!

You can also use speech-to-text! Start listening!

Use classifier trained with..

Twitter dataset

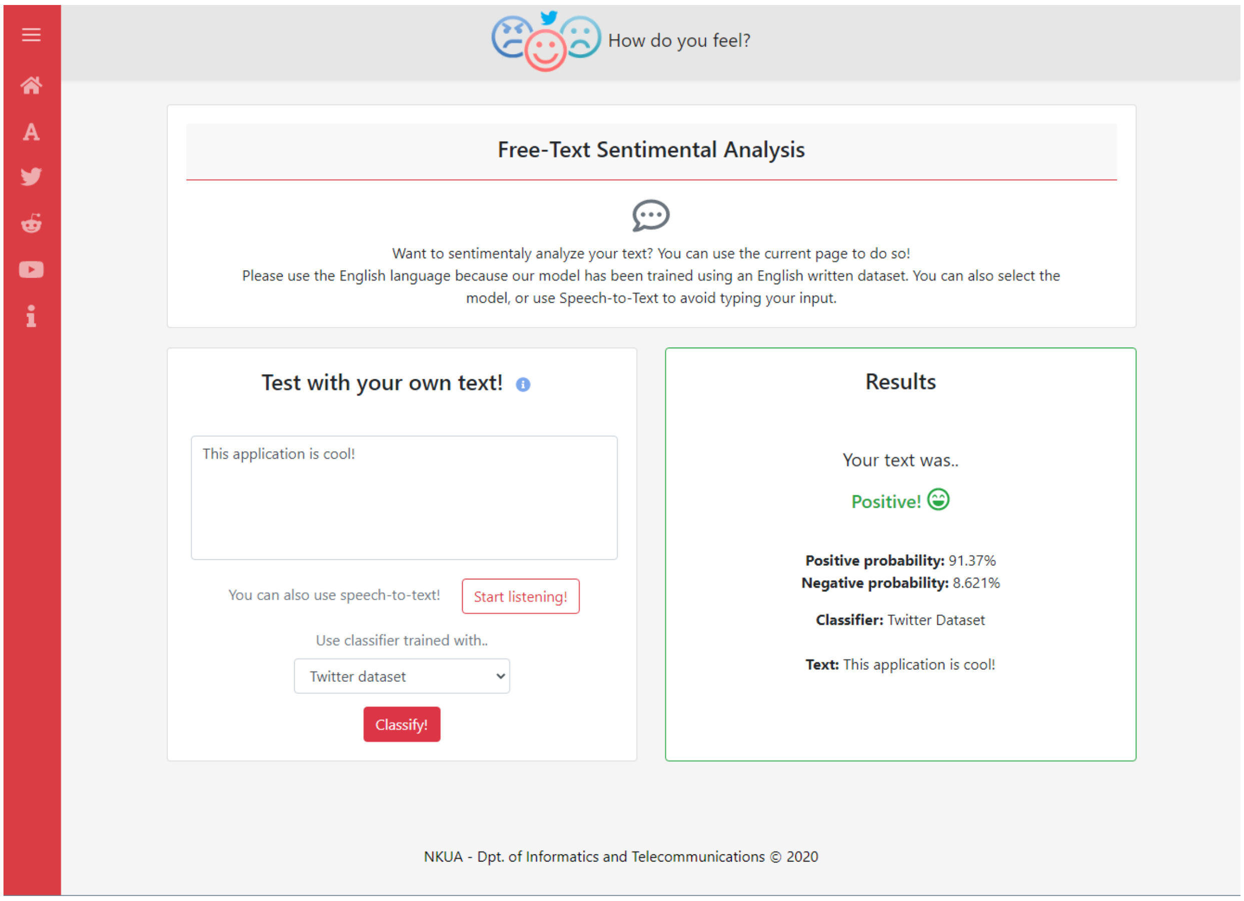
Classify!

Results

Your results will appear here..

NKUA - Dpt. of Informatics and Telecommunications © 2020

Figure 20: Free-text sentiment analysis layout


How do you feel?

Free-Text Sentimental Analysis

Want to sentimentaly analyze your text? You can use the current page to do so!
Please use the English language because our model has been trained using an English written dataset. You can also select the model, or use Speech-to-Text to avoid typing your input.

Test with your own text!

This application is cool!

You can also use speech-to-text! Start listening!

Use classifier trained with..

Twitter dataset

Classify!

Results

Your text was..

Positive!

Positive probability: 91.37%
Negative probability: 8.621%

Classifier: Twitter Dataset

Text: This application is cool!

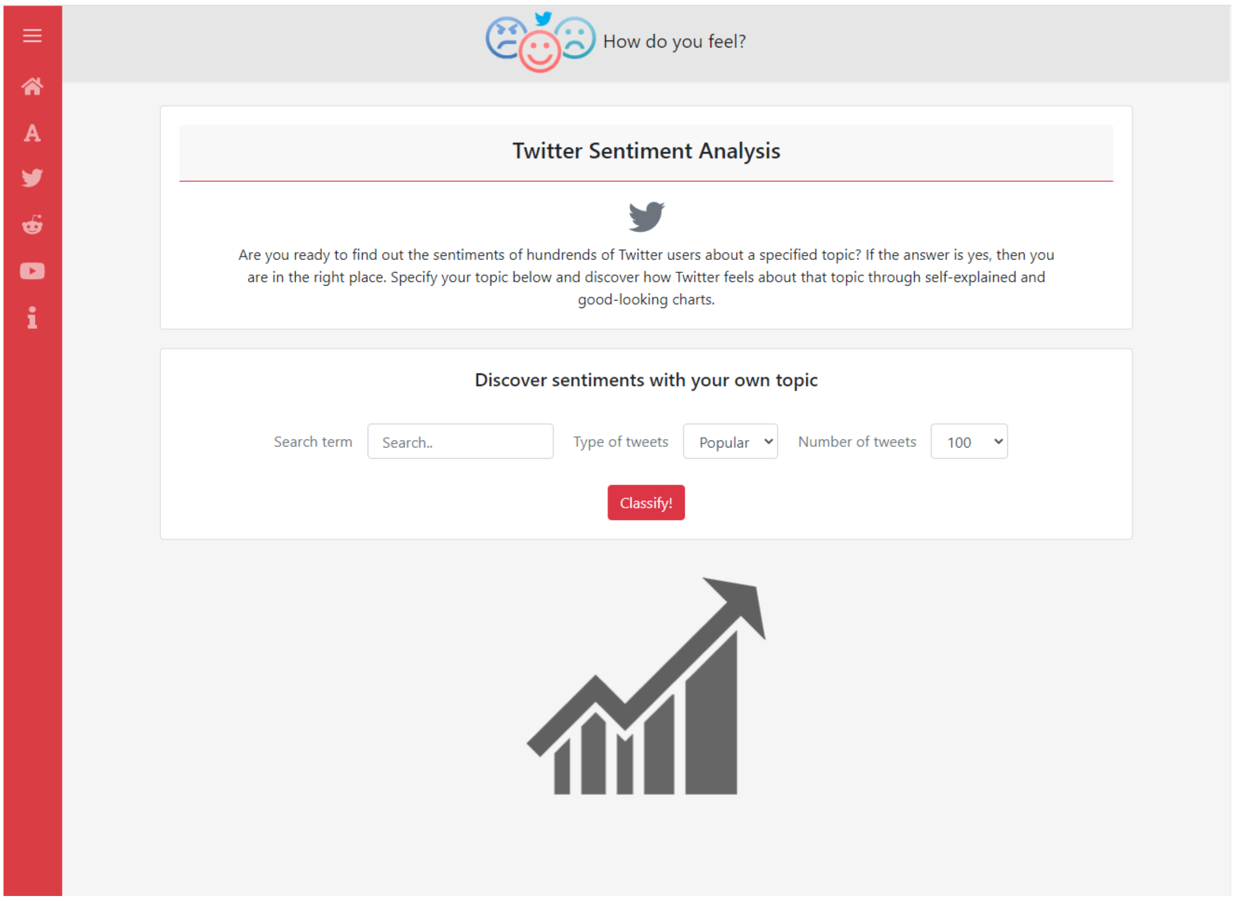
NKUA - Dpt. of Informatics and Telecommunications © 2020

Figure 21: Free-text sentiment analysis with result

7.4 Twitter

The Twitter integration page, accessible through the /twitter URI, provides three input fields, one textbox and two dropdown menus, which correspond to the respective fields of the request body of the /api/twitter/ endpoint. The textbox is for the query term input, while the one dropdown menu offers the three selections of the type of tweets to retrieve (popular, mixed and recent). The second option offers a small selection of preset comment numbers. The dropdown menus in this page, as well as in the following pages_(Reddit and Youtube), are valuable since they make the room of error of the request body invalid, offering only the correct values to the user. After the user inputs the parameters of his choice, the Axios module calls the /api/twitter/ method with the appropriate request body. When the results are yielded, four charts are created. These charts are created using the ChartJS library [81] with the data provided by the chartData object of the API response. The charts represent the number of tweets per sentiment, the number of “likes” and retweets per sentiment, the verified and non-verified users per sentiment, as well as the most tweeted from locations.

All of the charts, regardless of their type (pie or bar) are rendered with JavaScript, giving the user the ability to interact with them. A click on the labels can hide a specific class from the chart, while placing the mouse cursor on a particular part of the graph displays the value of the chart’s segment. Below the charts, there is a toggle button that reveals an array of all the tweets that were analyzed, essentially being the data that is visualized by the charts above. This array is populated by the data residing in the twitterData object of the request.


How do you feel?

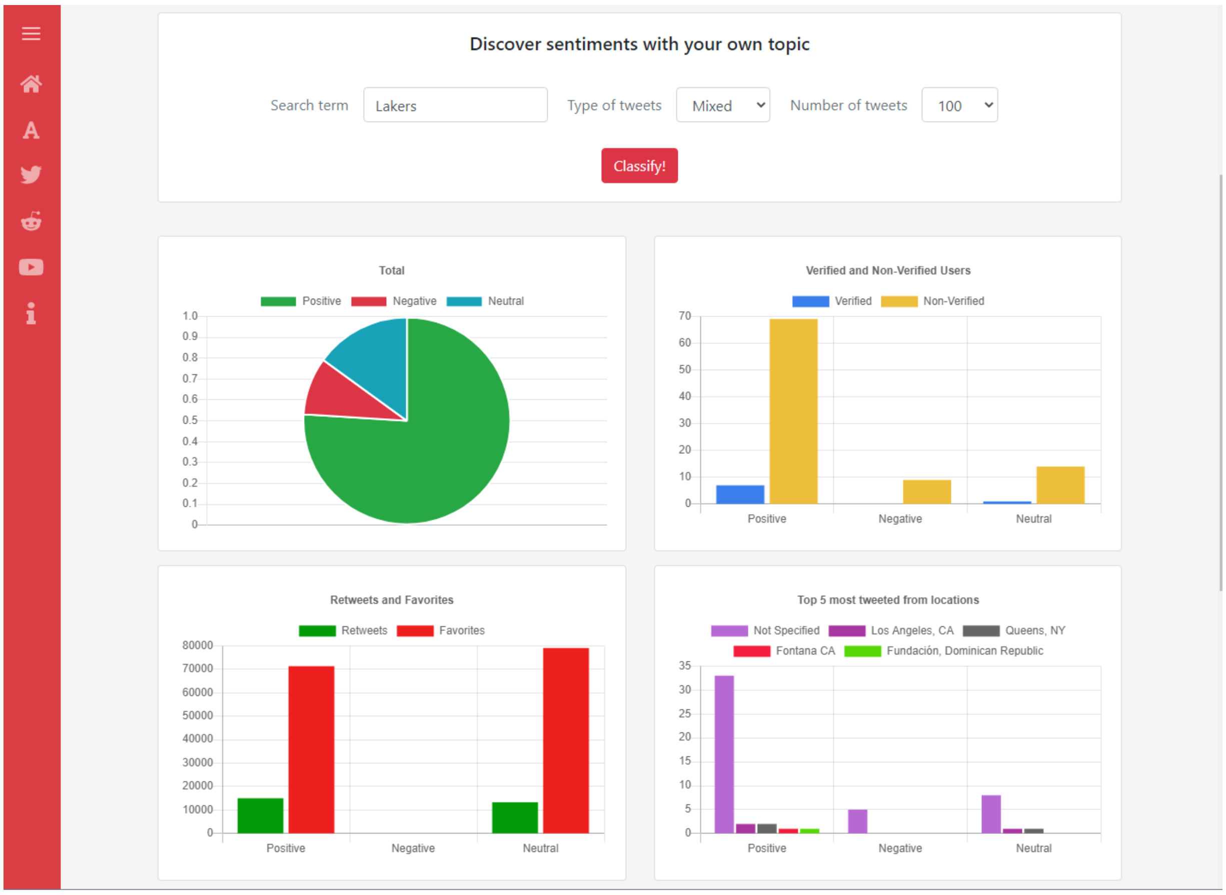
Twitter Sentiment Analysis

Are you ready to find out the sentiments of hundrends of Twitter users about a specified topic? If the answer is yes, then you are in the right place. Specify your topic below and discover how Twitter feels about that topic through self-explained and good-looking charts.

Discover sentiments with your own topic

Search term: Search..
Type of tweets: Popular
Number of tweets: 100

Figure 22: Twitter layout

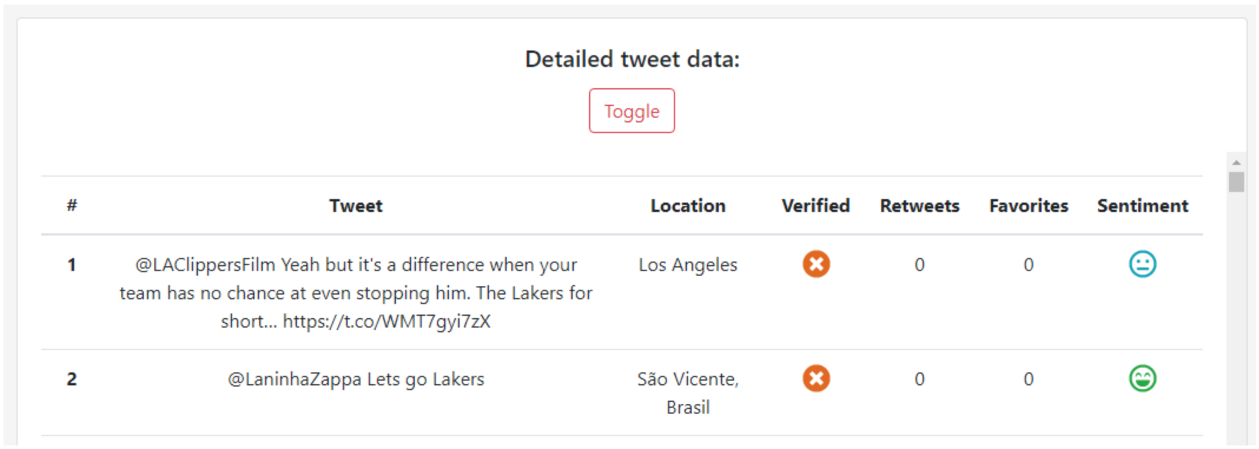


Discover sentiments with your own topic

Search term: Lakers Type of tweets: Mixed Number of tweets: 100

Classify!

Figure 23: Twitter results



Detailed tweet data:

Toggle

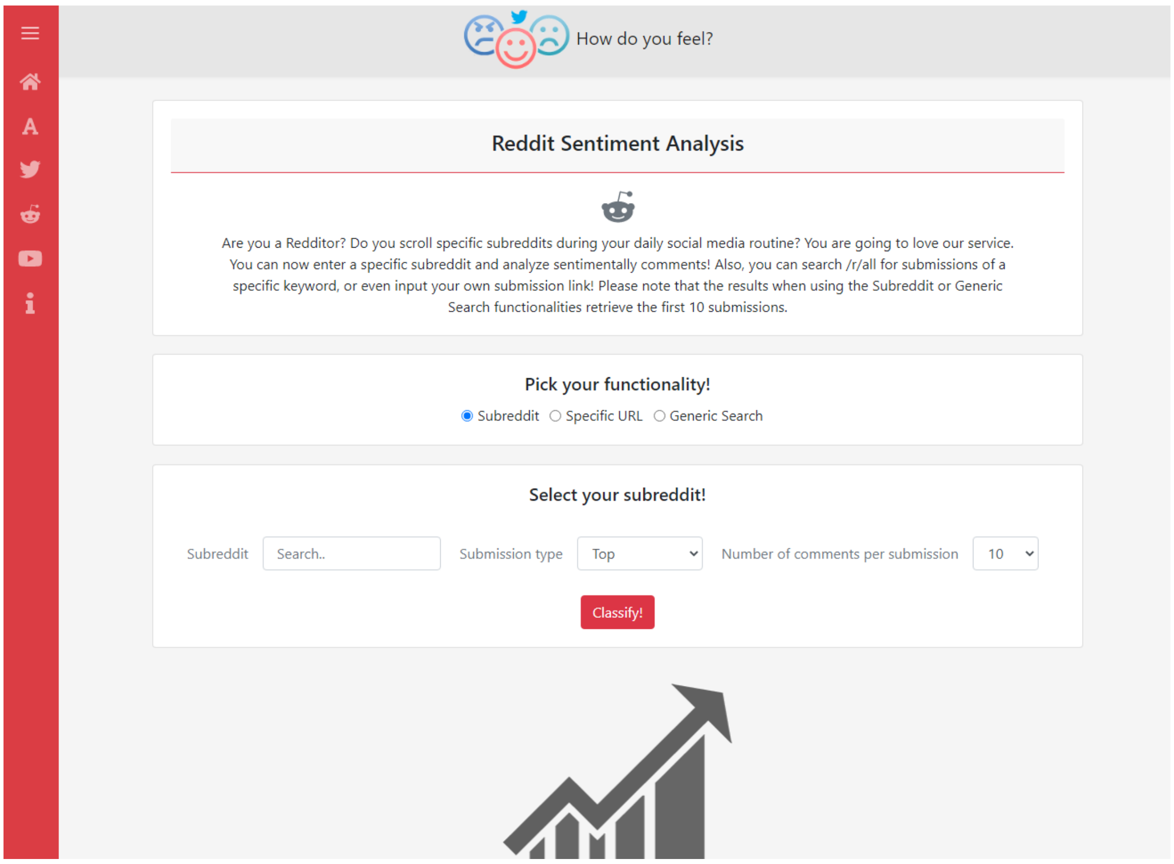
#	Tweet	Location	Verified	Retweets	Favorites	Sentiment
1	@LAClippersFilm Yeah but it's a difference when your team has no chance at even stopping him. The Lakers for short... https://t.co/WMT7gyi7zX	Los Angeles	✖	0	0	😐
2	@LaninhaZappa Lets go Lakers	São Vicente, Brasil	✖	0	0	😄

Figure 24: Twitter detailed results

7.5 Reddit

The /reddit URL of the platform offers three different usages, which correspond to the respective endpoints and request bodies, similarly to the Twitter page. A radio button swaps the inputs to the functionality of the user’s choice. This allows the possibility to perform sentiment analysis on the first 10 submissions of (i) a specific subreddit (/api/redditSub/) or (ii) a specific submission via URL (/api/redditURL/) or (iii) explore Reddit as a whole and extract the first 10 submissions returned for a specific keyword throughout its entirety (/api/redditSearch/).

The charts illustrate the following data: the number of comments per sentiment, the mean number of upvotes and replies, the mean link and comment “karma” of the authors and, finally, if the users are Reddit employees or Reddit gold owners. The data contained in the redditData object are displayed in the array below the charts, just like in the case of Twitter.



Reddit Sentiment Analysis

Are you a Redditor? Do you scroll specific subreddits during your daily social media routine? You are going to love our service. You can now enter a specific subreddit and analyze sentimentally comments! Also, you can search /r/all for submissions of a specific keyword, or even input your own submission link! Please note that the results when using the Subreddit or Generic Search functionalities retrieve the first 10 submissions.

Pick your functionality!

[✓ Subreddit] Specific URL Generic Search

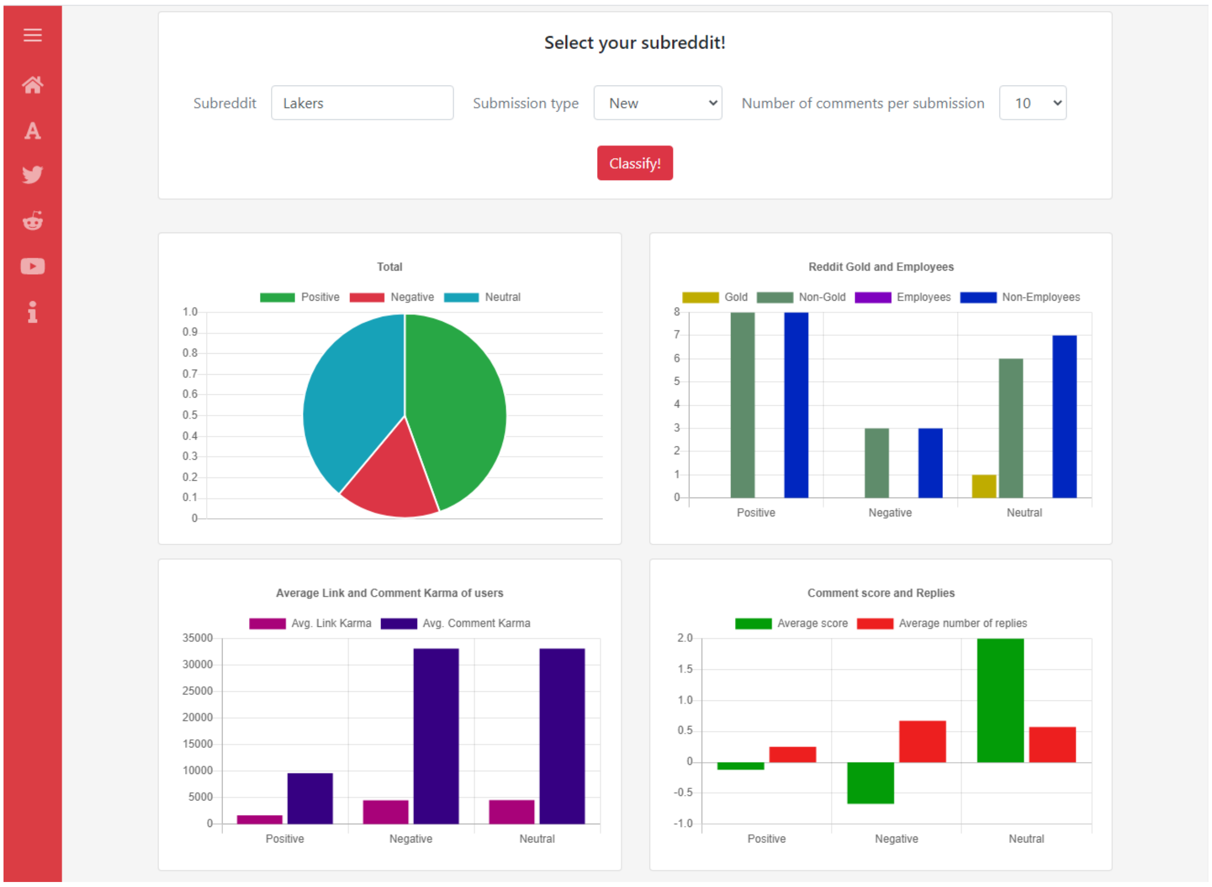
Select your subreddit!

Subreddit: Search..

Submission type: Top

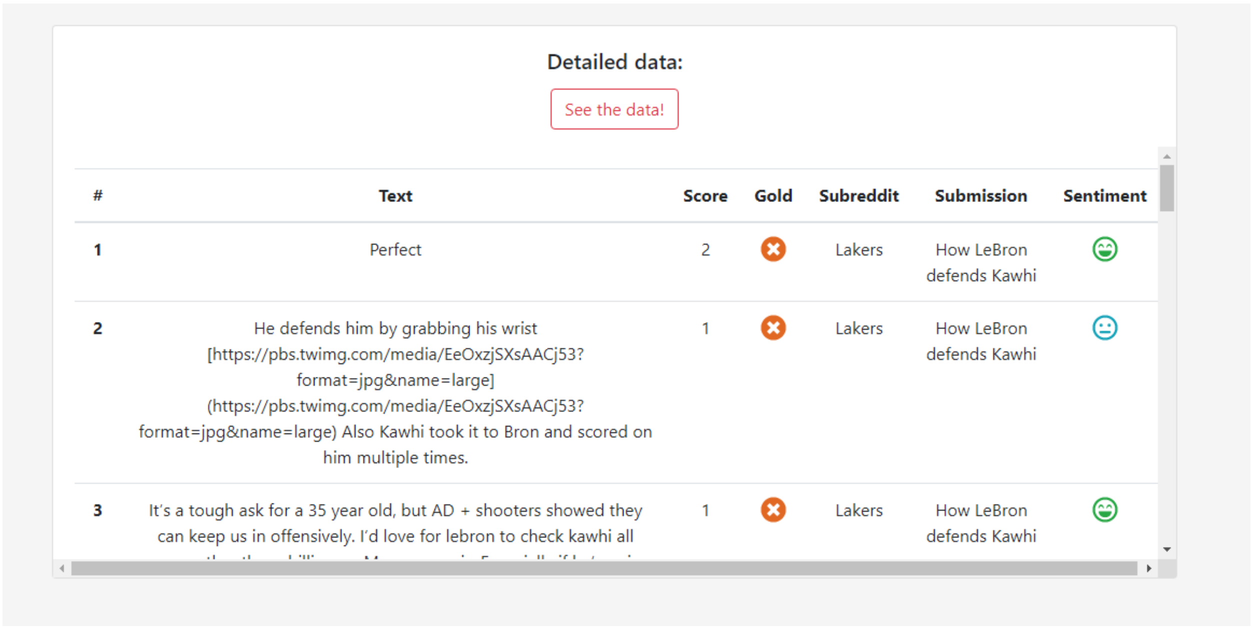
Number of comments per submission: 10

Figure 25: Reddit layout


Select your subreddit!

Subreddit: Lakers | Submission type: New | Number of comments per submission: 10

Figure 26: Reddit results


Detailed data:

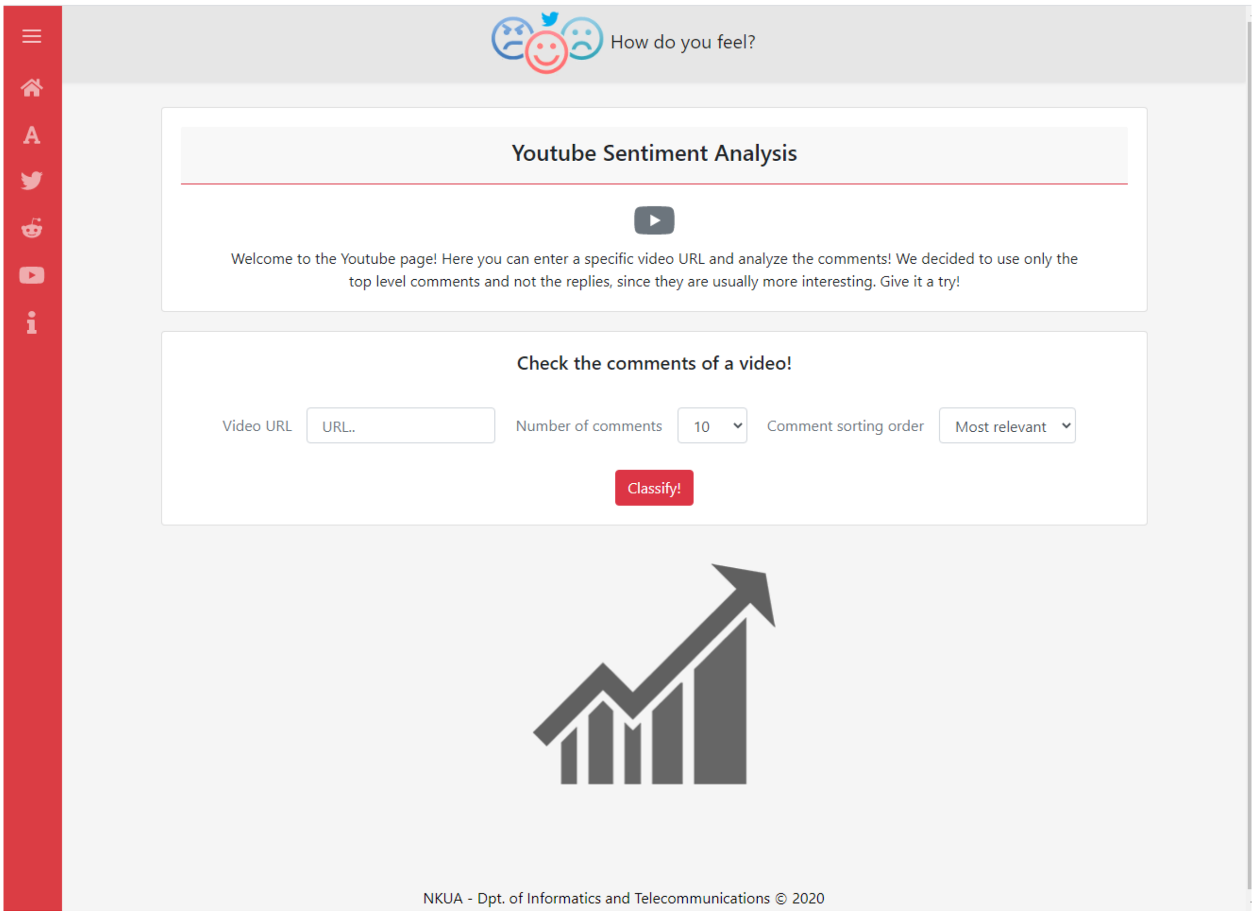
#	Text	Score	Gold	Subreddit	Submission	Sentiment
1	Perfect	2	✖	Lakers	How LeBron defends Kawhi	😄
2	He defends him by grabbing his wrist https://pbs.twimg.com/media/EeOxzjSXsAACj53?format=jpg&name=large Also Kawhi took it to Bron and scored on him multiple times.	1	✖	Lakers	How LeBron defends Kawhi	😐
3	It's a tough ask for a 35 year old, but AD + shooters showed they can keep us in offensively. I'd love for lebron to check kawhi all	1	✖	Lakers	How LeBron defends Kawhi	😄

Figure 27: Reddit detailed results

7.6 Youtube

The Youtube (/youtube) page functions in the same manner as the previous pages. It takes as inputs the three parameters that satisfy the request body of the /api/youtube/ endpoint, which are: a Youtube video URL, the number of comments to be fetched and the comment sorting method. The charts display the data collected by the

Application Programming Interface, which is a total number of comments per sentiment pie chart, the average “likes” per sentiment and information regarding the author’s channels, like the average subscriber count, video view count and video count per sentiment. In a similar manner to the other three functionalities, the youtubeData object is displayed at the bottom of the page.


How do you feel?

Youtube Sentiment Analysis

Welcome to the Youtube page! Here you can enter a specific video URL and analyze the comments! We decided to use only the top level comments and not the replies, since they are usually more interesting. Give it a try!

Check the comments of a video!

Video URL: URL..
Number of comments: 10
Comment sorting order: Most relevant

Classify!

NKUA - Dpt. of Informatics and Telecommunications © 2020

Figure 28: Youtube layout

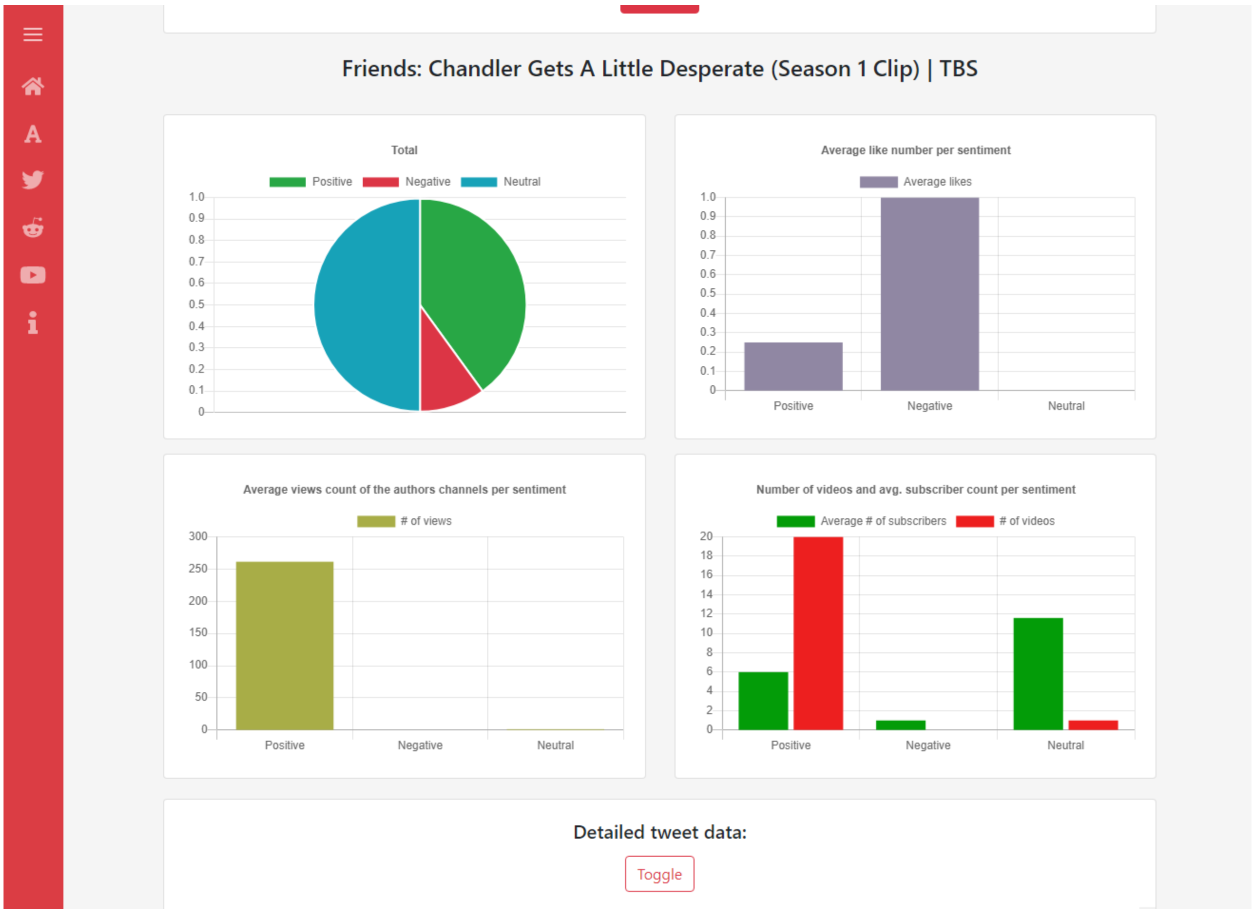
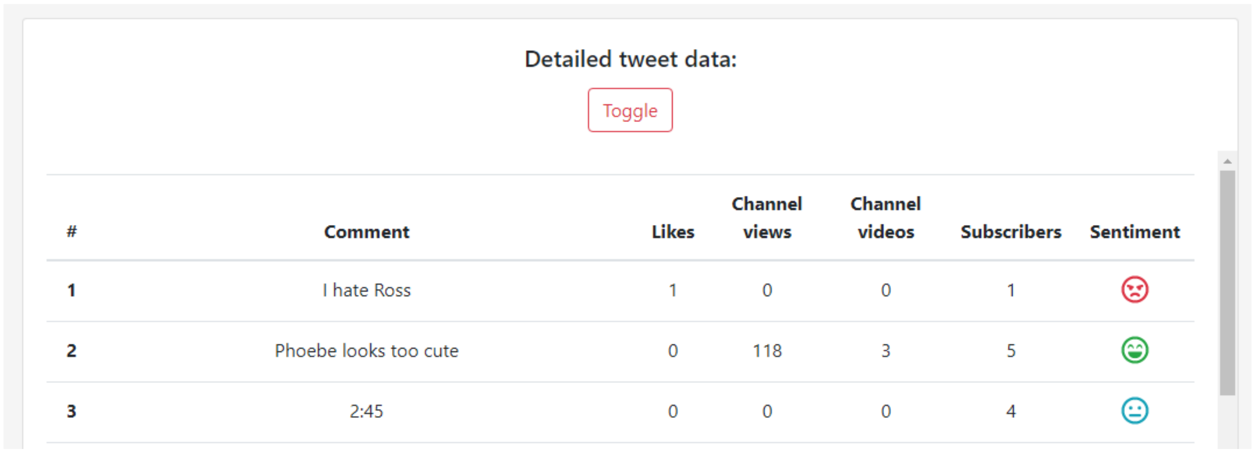

Friends: Chandler Gets A Little Desperate (Season 1 Clip) | TBS
Total
Positive
Negative
Neutral
Average like number per sentiment
Average likes
Average views count of the authors channels per sentiment
# of views
Number of videos and avg. subscriber count per sentiment
Average # of subscribers
# of videos
Detailed tweet data:
Toggle

Figure 29: Youtube results



Detailed tweet data:

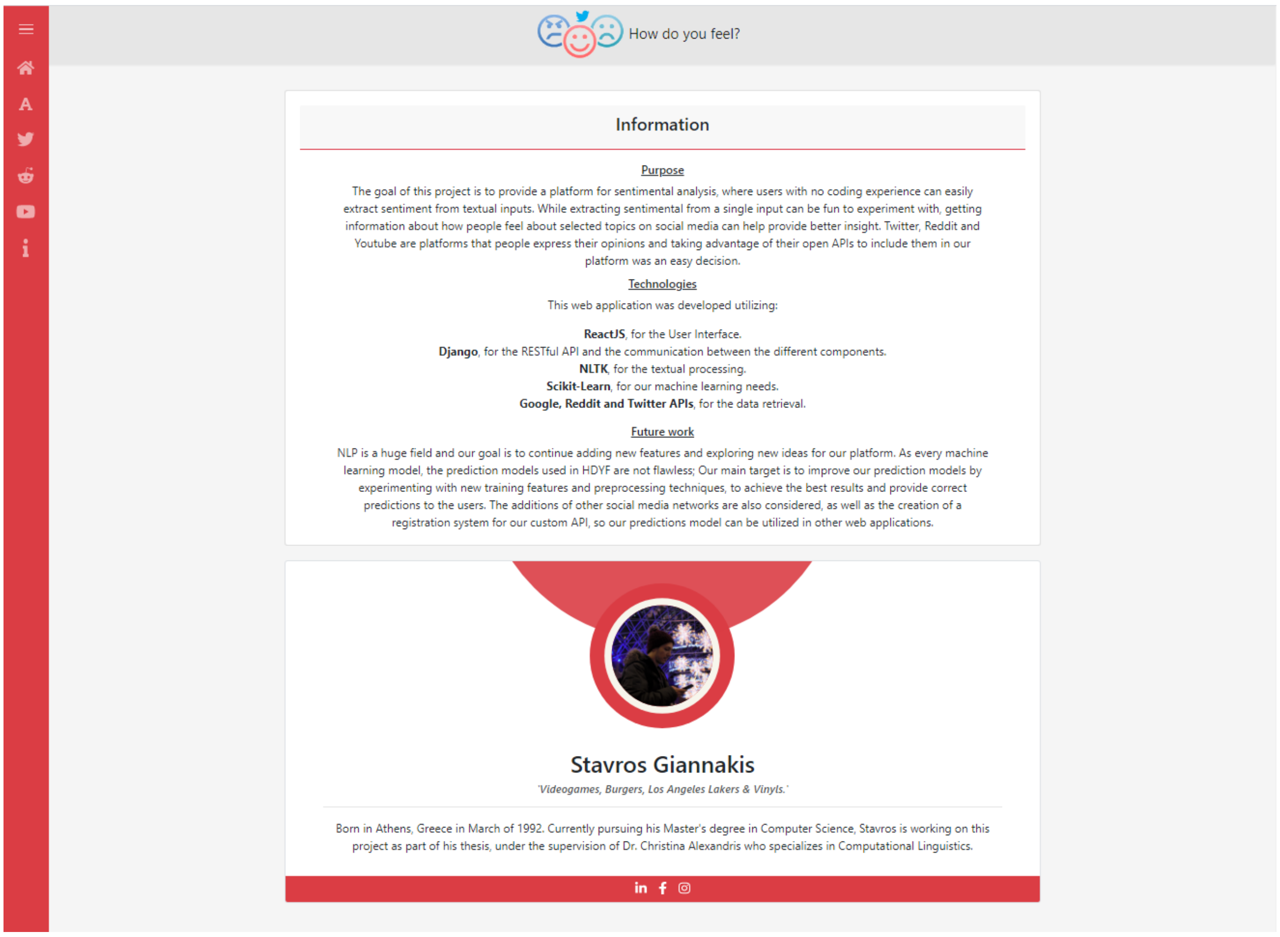
Toggle

#	Comment	Likes	Channel views	Channel videos	Subscribers	Sentiment
1	I hate Ross	1	0	0	1	😡
2	Phoebe looks too cute	0	118	3	5	😄
3	2:45	0	0	0	4	😐

Figure 30: Youtube detailed results

7.7 About

The last page of the platform, the “About” page, is a place to display generic information about the scope of the project and the tools used in the development process. This information is the programming languages and frameworks used for the implementation of the application, as well as some ideas regarding the future of the platform.



Information

Purpose

The goal of this project is to provide a platform for sentimental analysis, where users with no coding experience can easily extract sentiment from textual inputs. While extracting sentimental from a single input can be fun to experiment with, getting information about how people feel about selected topics on social media can help provide better insight. Twitter, Reddit and Youtube are platforms that people express their opinions and taking advantage of their open APIs to include them in our platform was an easy decision.

Technologies

This web application was developed utilizing:

ReactJS, for the User Interface.
Django, for the RESTful API and the communication between the different components.
NLTK, for the textual processing.
Scikit-Learn, for our machine learning needs.
Google, Reddit and Twitter APIs, for the data retrieval.

Future work

NLP is a huge field and our goal is to continue adding new features and exploring new ideas for our platform. As every machine learning model, the prediction models used in HDYF are not flawless; Our main target is to improve our prediction models by experimenting with new training features and preprocessing techniques, to achieve the best results and provide correct predictions to the users. The additions of other social media networks are also considered, as well as the creation of a registration system for our custom API, so our predictions model can be utilized in other web applications.

Stavros Giannakis

'Videogames, Burgers, Los Angeles Lakers & Vinyls.'

Born in Athens, Greece in March of 1992. Currently pursuing his Master's degree in Computer Science, Stavros is working on this project as part of his thesis, under the supervision of Dr. Christina Alexandris who specializes in Computational Linguistics.

Figure 31: About layout

7.8 Summary

Interfaces play a critical role in offering a satisfying experience to the user, with short response times and “clean” looking pages with smooth transitions. In this Chapter, we presented the ‘How do you feel?’ user interface, developed using ReactJS and JavaScript and enriched with CSS styling and Bootstrap.

We discussed about all the different pages that the web platform offers and the role that each page serves, tying them to the appropriate and corresponding Application Programming Interface endpoint presented in the previous Chapter (Chapter 6).

7.9 Adaptations for Greek Data

‘How do you feel?’ is a Sentiment Analysis platform and when the inclusion of different languages comes in question, the crucial feature is the machine learning model and the back end of the application. As previously described, the interface is synonymous with the user experience, which makes it equally important to the platform as the Application Programming Interface.

For the application’s interface to adapt to the Greek language, the most important feature is the possibility of the platform‘s availability in Greek. This can be achieved with the straight-forward solution of importing a page translator and localizing the web application in Greek, offering the menus and texts into the Greek language. This is considered a significant and important improvement, since many users that aim to use the platform to analyze Greek textual data may not be fluent in the English language, making the use of the platform challenging. An obvious solution for avoiding the erroneous translations would be to translate the texts of the web application and hard code them into the source files, which would be worth the extra effort.

Another significant addition would be the implementation of the speech-to-text feature to include the Greek language. This is the most challenging feature, due to the fact that the Web Speech API used in the present platform does not support the Greek language, so the development has to be made from scratch, or as an extension to the Web Speech API.

Finally, for the front-end input data to correspond to the API endpoints request inputs, the appropriate modifications will have to be made. For example, if the language of the text is decided to be inputted by the user, a new dropdown menu has to be added to the pages of the application, prompting the user to enter the language of their input.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

In the context of the present Thesis, we created three Sentiment Analysis models based on two popular datasets and their combination, with standard preprocessing techniques and parametrizations (1). These models offer the satisfying performance of an around 80% accuracy. We developed a full stack application where the end-user can interact with these three models, analyzing the polarity and extracting sentiment from any textual data of choice (2), offering, in addition, the opportunity to extract a large quantity of data from popular online social networks (3).

The basic features of the application presented in this Thesis are also the main benefits of the platform: The custom Sentiment Analysis models target to provide accurate results for two different document categories, namely tweets and movie reviews (a), while the user can also retrieve textual data for Sentiment Analysis from a variety of social media platforms (b). These features are offered to the end user via an easy to use and minimal user interface (c).

Specifically, in Chapter 2 we presented selected examples of related research and work that formed the basis of the approach chosen for the design, development and implementation of the proposed Sentiment Analysis project and application: the selection of algorithms, preprocessing techniques and feature extraction methods while developing our custom sentiment analysis models. Chapter 3 presented general information on the field of Computational Linguistics, as well as Natural Language Processing and Sentiment Analysis as a subsection of NLP, discussing main targets and goals, as well as common issues and problems.

In Chapter 4 we described the ‘How do you feel?’, the platform and project that this Thesis concerns. We presented the main goals and the features that differentiate this application from the already existing Sentiment Analysis implementations, as well as the target audiences and the datasets used for the development of the models. Chapter 5 explains the sentiment analysis machine learning models’ details by presenting the algorithms and techniques used for the development, with statistics and evaluations of the produced models.

In Chapter 6, we discuss the RESTful Application Programming Interface developed for the communication between the interface and the prediction models and present the custom endpoint functions and the data retrieval techniques when communicating with the integrated external social network APIs. Finally, in Chapter 7 describe and demonstrate the interface of the application, showing the features and the way the user can interact with the platform.

Throughout the Thesis, we also discussed the difficulties that emerge when trying to work with Greek data. While the development of a sentiment analysis model for the Greek language is the appropriate and optimal solution, other suggestions have been made to tackle the Greek inputs issue, such as translating the text to the English language using a machine translation model and applying the sentiment analysis techniques onto the translated data.

Many applications nowadays use textual data to extract valuable information for a wide variety of topics. Machine Learning and Natural Language Processing are used in a variety of tasks, both in industry and in various sectors, as well as in customized applications for personal usage. Sentiment Analysis is considered one of the challenging fields of Natural Language Processing, due to the fact that it can be expanded beyond the standard classification problem and used in applications like the one presented in this project to gather information. Platforms like the proposed “How do you feel” application are intended for a broad user group, including the general public, without them having to interact with a command line terminal. The targeted user group also includes students, helping them get a grasp of the basics of how machine learning works.

8.2 Future Work

“How do you feel” is just an entry level application in Sentiment Analysis and can be expanded in various, interesting and truly valuable ways.

Regarding the prediction models, extensive tuning of the hyper-parameters would provide beneficial improvements, as well as the inclusion of other datasets in the training procedure. Of course, as discussed earlier, the addition of more languages would be an amazing inclusion, with Greek being a priority, offering the opportunity to people speaking different languages to take advantage of this application. Finally, the introduction of deep neural networks instead of machine learning models would be a great challenge, with a neural network-based sentiment analysis application delivering even better results in similar projects.

As far as the web platform is concerned, some interesting extensions would be the integration of even more social media networks such as Facebook and Instagram. Another important update would be the ability to upload files for Sentiment Analysis, instead of just text in the free-text segment of the website. The API of the application could also be offered to other platforms, to take advantage and utilize the prediction models in their context.

LIST OF TRANSLATIONS

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Natural Language Processing	Επεξεργασία Φυσικής Γλώσσας
Sentiment Analysis	Ανάλυση Συναισθήματος
Opinion Mining	Εξόρυξη Γνώμης
Machine Learning	Μηχανική Μάθηση

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
BOW	Bag of Words
CL	Computational Linguistics
CS	Computer Science
HCI	Human Computer Interaction
IR	Information Retrieval
ML	Machine Learning
MT	Machine Translation
NLP	Natural Language Processing
OM	Opinion Mining
SA	Sentiment Analysis
TF-IDF	Term Frequency – Inverse Document Frequency
UI	User Interface

APPENDIX A. INSTALLATION GUIDE

To execute “How do you feel” on a computer locally, multiple packages and libraries need to be installed. Specifically:

1. Depending on the operating system that the application will be executed, the respective versions of NodeJS and Python with PIP, version >= 3.8 need to be installed.
2. With the required frameworks and programming language interpreters installed:
    a. To setup and execute the frontend part of the application, the “ThesisFront-master.zip” compressed file needs to be unzipped, typing “npm install” in a command line terminal to download the required dependencies and “npm start” to load and start the project.
    b. For the backend component of the project, after installing Python and unzipping the “ThesisBack-master.zip” file, the required Python modules are included in the “requirements.txt” file and can be installed collectively via the “pip install -r requirements.txt” command. Then, the machine learning models should be retrieved from the following repository https://github.com/cgs23/HowDoYouFeelModels/ and placed into the /classification/models directory of the project.To execute and start the API, the “python manage.py runserver” should be used.

REFERENCES

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow and R. Passonneau, "Sentiment Analysis of Twitter Data," 2011.

[2] P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," 2010. [Online]. Available: https://www.researchgate.net/publication/220746311_Twitter_as_a_Corpus_for_Sentiment_Analysis_and_Opinion_Mining.

[3] F. Colace, M. d. Santo and L. Greco, "A Probabilistic Approach to Tweets' Sentiment Classification," 2013. [Online]. Available: https://www.researchgate.net/publication/256472988_A_Probabilistic_Approach_to_Tweets'_Sentiment_Classification.

[4] A. Go, L. Huang and R. Bhayani, "Twitter Sentiment Analysis," 2009. [Online]. Available: https://nlp.stanford.edu/courses/cs224n/2009/fp/3.pdf.

[5] C. Troussas, M. Virvou, K. Espinosa, K. Llaguno and J. Caro, "Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning," 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6623713.

[6] G. Paltoglou and M. Thelwall, "https://dl.acm.org/doi/abs/10.1145/2337542.2337551," 2012. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2337542.2337551.

[7] M. Yasen and S. Tedmori, "Movies Reviews Sentiment Analysis and Classification," 2019. [Online]. Available: https://www.researchgate.net/publication/332321070_Movies_Reviews_Sentiment_Analysis_and_Classification.

[8] W. Kasper and M. Vela, "Sentiment Analysis for Hotel Reviews," 2012. [Online]. Available: https://www.researchgate.net/publication/275955836_Sentiment_Analysis_for_Hotel_Reviews.

[9] M. Welling, "A First Encounter with Machine Learning," 2011. [Online]. Available: https://www.ics.uci.edu/~welling/teaching/ICS273Afall11/IntroMLBook.pdf.

[10] A. Dey, "Machine Learning Algorithms: A Review," 2016. [Online]. Available: http://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit2016070332.pdf.

[11] T. Joutou and K. Yanai, "A Food Image Recognition System With Multiple Kernel Learning," 2009. [Online]. Available: https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICIP-2009/pdfs/0000285.pdf.

[12] S. Maruf and G. Haffari, "Document Context Neural Machine Translation with Memory Networks," 2018. [Online]. Available: https://www.aclweb.org/anthology/P18-1118.pdf.

[13] S. Loussaief and A. Abdelkrim, "Machine learning framework for image classification," 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7939841.

[14] S. Garg and N. Verma, "Study of Sentiment Classification Techniques," 2018. [Online]. Available: https://www.researchgate.net/publication/332343554_Study_of_Sentiment_Classification_Techniques.

[15] L. Maaten and E. Postma, "Dimensionality Reduction: A Comparative," 2009. [Online]. Available: https://members.loria.fr/moberger/Enseignement/AVR/Exposes/TR_Dimensiereductie.pdf.

[16] L. Schubert, "Stanford Encyclopedia of Philosophy," March 2020. [Online]. Available: https://plato.stanford.edu/archives/spr2020/entries/computational-linguistics/.

[17] V. S. Sakthi, "Applications of Computational Linguistics toLanguage Studies: An Overview," 2017. [Online]. Available: https://www.academia.edu/33152118/Applications_of_Computational_Linguistics_to_Language_Studies_An_Overview.

[18] H. Ney and F. Och, "The Alignment Template Approach to Statistical Machine Translation," 2004. [Online]. Available: https://www.aclweb.org/anthology/J04-4002.pdf.

[19] T. Bijimol and A. Johnt, "A Study of Machine Translation Methods An Analysis of Malayalam Machine Translation Systems," 2014. [Online]. Available: https://www.researchgate.net/publication/271689877_A_Study_of_Machine_Translation_Methods_An_Analysis_of_Malayalam_Machine_Translation_Systems.

[20] The Association for Computational Linguistics, "The Association for Computational Linguistics," February 2005. [Online]. Available: http://www.aclweb.org/archive/misc/what.html.

[21] S. Saad and B. Saberi, "Sentiment Analysis or Opinion Mining: A Review," 2017. [Online]. Available: https://www.researchgate.net/publication/320762707_Sentiment_Analysis_or_Opinion_Mining_A_Review.

[22] S. Sangar and D. Gupta, "Unsupervised Genre-Based Multidomain Sentiment Lexicon Learning Using Corpus-Generated Polarity Seed Words," 2020. [Online]. Available: https://www.researchgate.net/publication/342499084_Unsupervised_Genre-Based_Multidomain_Sentiment_Lexicon_Learning_Using_Corpus-Generated_Polarity_Seed_Words.

[23] P. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," 2002. [Online]. Available: https://nrc-publications.canada.ca/eng/view/accepted/?id=4bb7a0c8-9d9b-4ded-bcf6-fdf64ee28ccc.

[24] D. Jurafsky and J. Martin, Speech and Language Processing, 2018.

[25] B. Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer, 2009.

[26] J. Carbonell, "Subjective understanding: Computer models of belief systems," 1979. [Online]. Available: https://catalogue.nla.gov.au/Record/3286892.

[27] J. Carbonell, "Subjective understanding, computer models of belief systems," 1981. [Online]. Available: http://library.deakin.edu.au/record=b1163254.

[28] V. Hatzivassilogkou and K. McKeown, "Predicting the semantic orientation of adjectives," 1998. [Online]. Available: https://dl.acm.org/doi/10.3115/976909.979640.

[29] S. Vaithyanathan, B. Pang and L. Lee, "Thumbs up? Sentiment Classification using Machine Learning Techniques," 2002. [Online]. Available: https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf.

[30] M. V. Mäntylä and D. Graziotin, "The Evolution of Sentiment Analysis - A Review of Research Topics, Venues and Top Cited Papers," [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1612/1612.01556.pdf.

[31] A. D'Andrea and F. Ferri, "Approaches, Tools and Applications for Sentiment Analysis Implementation," 2015. [Online]. Available: https://www.researchgate.net/profile/Patrizia_Grifoni/publication/283201292_Approaches_Tools_and_Applications_for_Sentiment_Analysis_Implementation/links/5687f9b908aebccc4e15508b/Approaches-Tools-and-Applications-for-Sentiment-Analysis-Implementation.pdf.

[32] F. Chaumartin, "A knowledge-based system for headline sentiment tagging," 2011. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00611242/document.

[33] N. Fernando, "Knowledge Based Approach for Concept Level Sentiment Analysis for Online Reviews," 2016. [Online]. Available: https://www.researchgate.net/publication/328189753_Knowledge_Based_Approach_for_Concept_Level_Sentiment_Analysis_for_Online_Reviews.

[34] P. Turney, "Measuring Praise and Criticism: Inference of Semantic Orientation from Association," 2003. [Online]. Available: https://www.researchgate.net/publication/200044323_Measuring_Praise_and_Criticism_Inference_of_Semantic_Orientation_from_Association.

[35] V. Hatzivassiloglou and K. McKeown, "Predicting the Semantic Orientation of Adjectives," 1997. [Online]. Available: https://www.aclweb.org/anthology/P97-1023/.

[36] J. Ortigosa-Hernández and J. Rodríguez, "Approaching Sentiment Analysis by Using Semi-supervised Learning of Multidimensional Classifiers," 2011. [Online]. Available: http://addi.ehu.es:8080/bitstream/handle/10810/4762/tr11-4.pdf?sequence=1&isAllowed=y.

[37] M. Huq, A. Ali and A. Rahman, "Sentiment Analysis on Twitter Data using KNN and SVM," 2017. [Online]. Available: https://pdfs.semanticscholar.org/05a8/78000170abcd0c6f8208080470858422e17c.pdf.

[38] Y. E. Cakra, "Stock price prediction using linear regression based on sentiment analysis," 2015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7415179.

[39] S. Mohhamad, "Challenges in Sentiment Analysis," 2017. [Online]. Available: https://www.researchgate.net/publication/316049100_Challenges_in_Sentiment_Analysis.

[40] M. Wiegand, A. Balahur, B. Roth, D. Klakow and A. Montoyo, "A Survey on the Role of Negation in Sentiment Analysis," 2010. [Online]. Available: https://www.aclweb.org/anthology/W10-3111.pdf.

[41] D. Maynard and M. Greenwood, "Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis," 2014. [Online]. Available: http://eprints.whiterose.ac.uk/130763/1/sarcasm.pdf.

[42] P. Bhattacharyya, A. Ekbal, R. Dhanush, Chauhan and D, "Sentiment and Emotion help Sarcasm? A Multi-task Learning: Framework for Multi-Modal Sarcasm, Sentiment and Emotion Analysis," 2020. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.401.pdf.

[43] Y. Wu and P. Jin, "Disambiguating Sentiment Ambiguous Adjectives," 2018. [Online]. Available: https://www.aclweb.org/anthology/S10-1014.pdf.

[44] A. Khan, "Context-Aware Spelling Corrector for Sentiment Analysis," 2014. [Online]. Available: https://www.researchgate.net/publication/284344945_Context-Aware_Spelling_Corrector_for_Sentiment_Analysis.

[45] M. Hussein, "A survey on sentiment analysis challenges," 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1018363916300071.

[46] R. Feldman, "Techniques and applications for sentiment analysis," 2013. [Online]. Available: https://dl.acm.org/doi/fullHtml/10.1145/2436256.2436274.

[47] A. Go, R. Bhayani and L. Huang, "Sentiment140," [Online]. Available: http://help.sentiment140.com/.

[48] A. Maas, D. Daly, P. Pham, D. Huang, A. Ng and C. Potts, "Large Movie Review Dataset," 2011. [Online]. Available: http://ai.stanford.edu/~amaas/data/sentiment/.

[49] M. Rahmannia and S. Triyono, "A Study of Google Translate Translations: An Error Analysis of Indonesian-to-English Texts," 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3456744.

[50] P. e. al., "Scikit-learn: Machine Learning in Python," JMLR 12, pp. 2825-2830, 2011.

[51] S. Walt, C. Colbert and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5725236/.

[52] T. Oliphant, "Python for Scientific Computing," 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4160250.

[53] S. E. L. a. E. K. Bird, Natural Language Processing with Python, O’Reilly Media Inc., 2011.

[54] A. Krouska, C. Troussas and M. Virvou, "The effect of preprocessing techniques on Twitter sentiment analysis," 2016. [Online]. Available: https://www.researchgate.net/publication/311755864_The_effect_of_preprocessing_techniques_on_Twitter_sentiment_analysis.

[55] V. Gurusamy and S. Kannan, "Preprocessing Techniques for Text Mining," 2014. [Online]. Available: https://www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining.

[56] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances," 2014. [Online]. Available: http://eprints.um.edu.my/13423/1/rp030_I3007.pdf.

[57] M. Asghar, A. Khan, S. Ahmad and F. Kundi, "A Review of Feature Extraction in Sentiment Analysis," 2014. [Online]. Available: http://www.statmt.org/OSMOSES/FeatureEx.pdf.

[58] A. V. Rozhnov and A. A. Melikhov, "Vectorizing textual data sources to decrease attribute space dimension," 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8109662.

[59] C. Qian and L. Kermode, "Effective Sentiment Analysis using a Bag of Words Representation," [Online]. Available: http://cjqian.github.io/docs/sentiment.pdf.

[60] S. Qaiser, "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents," 2018. [Online]. Available: https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents.

[61] F. Enríquez, F. L.Cruz, F. J. Ortega, C. G. Vallejo and J. A. Troyano, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1566253512000425.

[62] M. Setyawan, R. Awangga and S. Efendi, "Comparison Of Multinomial Naive Bayes Algorithm And Logistic Regression For Intent Classification In Chatbot," 2018. [Online]. Available: https://www.researchgate.net/publication/329817566_Comparison_Of_Multinomial_Naive_Bayes_Algorithm_And_Logistic_Regression_For_Intent_Classification_In_Chatbot.

[63] A. Kibriya, E. Frank, B. Pfahringer and G. Holmes, "Multinomial Naive Bayes for Text Categorization Revisited," 2004. [Online]. Available:

https://perun.pmf.uns.ac.rs/radovanovic/dmsem/cd/install/Weka/doc/pubs/2004/KibriyaAI04-MultinomialNBRevisited.pdf.

[64] D. Jurafsky and J. H. Martin, "Speech and Language Processing," 2018, pp. 65-68.

[65] H. Hamdan, P. Bellot and F. Bechet, "Lsislif: CRF and Logistic Regression for Opinion Target Extraction and Sentiment Polarity Analysis," 2015.

[66] P. Willett, "The Porter stemming algorithm: Then and now," 2006. [Online]. Available: https://www.researchgate.net/publication/33038304_The_Porter_stemming_algorithm_Then_and_now.

[67] Django Software Foundation, "Django," Django Software Foundation, 2013. [Online]. Available: https://djangoproject.com.

[68] "Django REST Framework," [Online]. Available: https://www.django-rest-framework.org/.

[69] F. Manca, "Django REST Framework API Key," 2018. [Online]. Available: https://florimondmanca.github.io/djangorestframework-api-key/.

[70] "Tweepy," [Online]. Available: https://www.tweepy.org/.

[71] Twitter, "Introduction to Tweet JSON," Twitter, [Online]. Available: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.

[72] PRAW, "PRAW Official Documentation," [Online]. Available: https://praw.readthedocs.io/.

[73] Google, "Google's Data API Documentation," [Online]. Available: https://developers.google.com/youtube/v3.

[74] Google, "Google API Client," [Online]. Available: https://github.com/googleapis/google-api-python-client.

[75] W. Cavnar and J. Trenkle, "N-Gram-Based Text Categorization," 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.3248&rep=rep1&type=pdf.

[76] F. Lopez, "langdetect," 2020. [Online]. Available: https://github.com/fedelopez77/langdetect.

[77] Google LLC, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," 2016. [Online]. Available: https://www.researchgate.net/publication/308646556_Google's_Neural_Machine_Translation_System_Bridging_the_Gap_between_Human_and_Machine_Translation.

[78] Facebook, "ReactJS," Facebook, [Online]. Available: https://reactjs.org/.

[79] "Bootstrap," [Online]. Available: https://getbootstrap.com/.

[80] "Axios," [Online]. Available: https://github.com/axios/axios.

[81] "ChartJS," [Online]. Available: https://www.chartjs.org/.