

UNIVERSITY OF OVIEDO



SCHOOL OF COMPUTER ENGINEERING

FINAL DEGREE PROJECT

“Inappropriate message detection system/Sistema de detección de mensajes inapropiados”

DIRECTOR: Cristian González García

CODIRECTOR: Vicente García Díaz

AUTHOR: Adrián Pérez Manso

Acknowledgments

To my family and friends.

To my dear friend Luis, for knowing when to act as a mentor and when to act as a friend.

To my grandmother, that could not see me graduating.

To my partner Marta, my constant support.

Resumen

La proliferación de las redes sociales y de espacios de debate en Internet ha alterado completamente nuestra percepción sobre conceptos como lo políticamente correcto o los límites del humor. Voces que antes no eran escuchadas debido a la falta de medios tienen ahora incluso más alcance que los medios de comunicación tradicionales. Aunque esta democratización de la información y las opiniones conllevan un gran avance, existe un crecimiento proporcional de número de comentarios los cuales tienen intenciones nocivas. Sumado a esto, la sensibilidad general de la Sociedad se ha afinado como consecuencia de que las nuevas voces que surgen puedan expresar y explicar qué comportamientos son incorrectos y por qué. Estos nuevos elementos propios del siglo XXI, ha provocado que muchas empresas busquen mecanismos para prevenir los comentarios que suscriban características ofensivas, tóxicas y de odio.

El objetivo de este proyecto es la implementación de un sistema capaz de clasificar un texto dentro de una escala que refleje cuán inapropiado es este, entendiéndose por inapropiado una propiedad basada en elementos de toxicidad, obscenidad, amenaza, insulto u apología al odio. A su vez, este trabajo tiene la finalidad de desarrollar un estudio comparativo de las posibles alternativas de implementación que tendría este sistema.

El programa ofrece la posibilidad de realizar una predicción del texto facilitado como entrada a través de dos métodos: el primero asigna al texto un valor de dos posibles (apropiado o inapropiado). Por otra parte, el segundo método permite diseccionar la predicción del texto en propiedades más específicas como las comentadas anteriormente.

Otro aspecto del sistema es el de ofrecer a una persona, con las credenciales pertinentes, entrenar el modelo nuevamente con conjuntos de datos introducidos por ella. Por último, ciertas operaciones de interés para el usuario como salvar archivos con las predicciones realizadas son planteadas.

En el proyecto se explora principalmente el uso de la inteligencia artificial y el aprendizaje automático, basados tanto en modelos matemáticos como en redes neuronales. En mayor detalle, el sistema se basa en una tarea de clasificación de texto, como puede ser a su vez la clasificación de sentimientos a partir de un texto. Asimismo, se investiga cómo se enmarca este sistema dentro de la dirección y planificación de proyectos.

Palabras Clave

Mensajes tóxicos, Discurso de odio, Clasificación de texto, Inteligencia artificial, Aprendizaje automático, Aprendizaje profundo, Procesamiento de lenguaje natural

Abstract

The proliferation of social networks and rooms of discussion on the Internet has completely altered our idea about concepts such as the politically correct and humour boundaries. The voices that were not heard before due to lack of means have these days even more reach than traditional media. Although this information and opinion democratization imply huge progress, there has been a proportional growth of number of comments with harmful intentions. Moreover, the general sensitivity of society has been polished as a consequence of the new voices that emerge and can express and explain what behaviours are incorrect and why. These new elements, intrinsic to the 21st century, have made some companies attempt to prevent those comments that subscribe offensive, toxic and hate characteristics.

The goal of this project is the implementation of a system capable of classifying a text within a scale that reflects how appropriate it is, assuming as inappropriate any message with toxic, obscene, threat, insult and/or identity hate elements. Furthermore, this piece of work has the intention of developing a thoughtful comparative study about the possible implementation alternatives which this system could have.

The program offers the possibility of doing a prediction of an input text via two methods: The first one assigns a value from two possible to the text (appropriate or inappropriate). On the other hand, the second method allows to dissect the prediction into more specific properties as the ones mentioned previously.

Another aspect of the system is that it grants the possibility to an individual, with its corresponding credentials, of training the model with new datasets as input. Finally, some user-friendly operations such as saving the predictions into a file are suggested.

This project explores mainly the use of artificial intelligence and machine learning, based both on statistical mathematic models and on artificial neural networks. More specifically, the system is enclosed on a text classification task, which includes a common field of work called Natural Language Processing (NLP). For instance, sentiment classification is a popular task inside this area. Likewise, this work does research about how this sort of system is crafted on project management and planification framework.

Keywords

Toxic messages, Hate speech, Text classification, Artificial Intelligence, Machine Learning, Deep Learning, Natural language processing

General index

Table of contents

CHAPTER 1. PROJECT REPORT	15
1.1 MOTIVATION SUMMARY, OBJECTIVES AND PROJECT SCOPE	15
1.2 SUMMARY OF ALL ASPECTS	16
CHAPTER 2. INTRODUCTION	17
2.1 JUSTIFICATION OF THE PROJECT	17
2.2 OBJECTIVES OF THE PROJECT	18
2.3 STUDY OF CURRENT SITUATION	19
2.3.1 <i>Alternatives evaluation</i>	19
CHAPTER 3. THEORETICAL ASPECTS	23
3.1 ARTIFICIAL INTELLIGENCE (AI).....	23
3.2 MACHINE LEARNING (ML).....	23
3.3 DEEP LEARNING (DL).....	24
3.4 LINEAR REGRESSION.....	24
3.5 LOGISTIC REGRESSION	25
3.6 STOCHASTIC GRADIENT DESCENT	25
3.7 NEURAL NETWORK	26
3.8 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)	27
3.9 PYTHON.....	27
3.10 SCIKIT LEARN	28
3.11 PYTORCH.....	28
3.12 TRANSFORMERS.....	28
CHAPTER 4. PROJECT PLANNING AND INITIAL BUDGETS	30
4.1 INITIAL PLANNING	30
4.2 INITIAL BUDGET	31
4.2.1 <i>Company definition</i>	31
4.2.2 <i>Costs budget</i>	34
4.2.3 <i>Client budget</i>	39
CHAPTER 5. ANALYSIS	41
5.1 SYSTEM DEFINITION.....	41
5.1.1 <i>System scope specification</i>	41
5.2 SYSTEM REQUIREMENTS	41
5.2.1 <i>Elicitation</i>	41
5.2.2 <i>System Actors Identification</i>	44
5.2.3 <i>Use cases specification</i>	45
5.3 SUBSYSTEMS IDENTIFICATION IN ANALYSIS PHASE	46

5.3.1	Subsystems description.....	47
5.3.2	Interfaces between systems description.....	47
5.4	INITIAL CLASS DIAGRAM IN ANALYSIS PHASE.....	47
5.4.1	Class diagram.....	47
5.4.2	Classes description.....	48
5.5	USE CASES ANALYSIS AND SCENARIOS.....	50
5.5.1	Change classification method.....	50
5.5.2	Detect inappropriate messages.....	51
5.5.3	Save results to a file.....	51
5.5.4	Correct predictions.....	52
5.5.5	Log in as administrator.....	52
5.5.6	Train models.....	53
5.6	SCENARIOS – USE CASES RELATION.....	54
5.7	USER INTERFACE ANALYSIS.....	54
5.7.1	Interface description.....	54
5.7.2	Interface behaviour description.....	60
5.7.3	Navigability diagram.....	61
5.8	TEST PLAN SPECIFICATION.....	61
5.8.1	Unitary testing.....	61
5.8.2	Integration testing.....	61
5.8.3	Usability testing.....	62
5.8.4	Performance testing.....	62
5.8.5	Use cases testing.....	62
CHAPTER 6.	SYSTEM DESIGN.....	66
6.1	SYSTEM ARCHITECTURE.....	66
6.1.1	Package diagram.....	66
6.1.2	Component diagram.....	68
6.2	CLASS DESIGN.....	68
6.2.1	Controller package.....	68
6.2.2	Authentication.....	70
6.2.3	Classifiers.....	70
6.2.4	Utils.....	71
6.2.5	Config.....	71
6.2.6	Domain.....	72
6.2.7	User interface.....	73
6.2.8	Global class diagram.....	73
6.3	INTERACTION DIAGRAMS.....	74
6.3.1	Detect inappropriate messages.....	74
6.3.2	Train model use case.....	75
6.3.3	Correct predictions use case.....	76
6.3.4	Log in as administrator use case.....	77
6.3.5	Change classification method use case.....	77
6.3.6	Save results to file use case.....	78
6.4	SYSTEM PERSISTENCE.....	78
6.4.1	Database design.....	78
6.5	INTERFACE DESIGN.....	79
6.5.1	Main window.....	79
6.5.2	Authentication window.....	80
6.5.3	Training window.....	80

6.5.4	<i>Confirmation window</i>	81
6.6	TEST PLAN TECHNICAL SPECIFICATION	81
6.6.1	<i>Unit testing</i>	82
6.6.2	<i>Integration tests</i>	85
6.6.3	<i>Usability tests</i>	85
6.6.4	<i>Accessibility tests</i>	88
6.6.5	<i>Performance tests</i>	88
CHAPTER 7.	SYSTEM IMPLEMENTATION	90
7.1	PROGRAMMING LANGUAGES	90
7.1.1	<i>Python</i>	90
7.2	TOOLS AND PROGRAMS USED FOR THE DEVELOPMENT OF THE SYSTEM	91
7.2.1	<i>Visual Studio Code</i>	91
7.2.2	<i>Git</i>	92
7.3	SYSTEM CREATION	92
7.3.1	<i>Binary model</i>	92
7.3.2	<i>Itemized model</i>	94
7.3.3	<i>Application and interface</i>	95
7.3.4	<i>Last modifications</i>	95
7.3.5	<i>Found issues</i>	95
7.3.6	<i>Detailed class description</i>	101
CHAPTER 8.	EVALUATION OF ALTERNATIVES	102
8.1	STATE OF ART SOLUTIONS AND STATISTICS	102
8.1.1	<i>Binary models</i>	102
8.1.2	<i>Multilabel models</i>	104
8.2	ATTEMPTED ALTERNATIVES	105
8.2.1	<i>Binary models</i>	105
8.2.2	<i>Multilabel models</i>	112
CHAPTER 9.	TESTING DEVELOPMENT	118
9.1	UNIT TESTS	118
9.2	USABILITY TESTS	121
9.2.1	<i>User profile ranking</i>	121
9.2.2	<i>Guided activities</i>	123
9.2.3	<i>Quick questions about the application</i>	125
9.2.4	<i>Tester survey</i>	128
9.3	ACCESSIBILITY TESTS	129
9.4	PERFORMANCE TESTS	130
CHAPTER 10.	SYSTEM MANUALS	132
10.1	INSTALLATION MANUALS	132
10.2	EXECUTION MANUALS	133
10.3	USER MANUAL	133
10.3.1	<i>Non-administrator user manual</i>	134
10.3.2	<i>Administrator manual</i>	141
10.4	PROGRAMMER MANUAL	145
10.4.1	<i>Add a new classifier</i>	146
10.4.2	<i>Change the interface</i>	146
CHAPTER 11.	CONCLUSIONS AND FUTURE WORK	148

11.1	CONCLUSIONS	148
11.2	EXTENSIONS.....	150
CHAPTER 12.	PROJECT PLANNING AND FINAL BUDGETS	151
12.1	FINAL PLANNING	151
12.2	FINAL BUDGET.....	152
12.2.1	<i>Costs budget</i>	152
CHAPTER 13.	BIBLIOGRAPHIC REFERENCES	158
CHAPTER 14.	ANNEXES	167
14.1	GLOSSARY	167
14.2	DELIVERED CONTENT IN ATTACHED FILE	168
14.2.1	<i>Contents</i>	168
14.2.2	<i>Executable code and installation</i>	169
14.3	SOURCE CODE	169
14.4	MEETING MINUTES.....	170

Figure Index

FIGURE 1 MULTILABEL CLASSIFICATION EXAMPLES.....	21
FIGURE 2 BINARY RELEVANCE TRANSFORMATION EXAMPLE	21
FIGURE 3 CLASSIFIER CHAIN TRANSFORMATION EXAMPLE.....	21
FIGURE 4 LABEL POWERSSET EXAMPLE	22
FIGURE 5 FORMULA OF A SIMPLE LINEAR REGRESSION	24
FIGURE 6 FIRST FORMULA OF LOGISTIC REGRESSION FOR ONE VARIABLE.....	25
FIGURE 7 SECOND FORMULA OF LOGISTIC REGRESSION FOR ONE VARIABLE.....	25
FIGURE 8 REPRESENTATION OF AN ARTIFICIAL NEURON.....	26
FIGURE 9 PYTHON LOGO	27
FIGURE 10 SCIKIT LEARN LOGO	28
FIGURE 11 PYTORCH LOGO.....	28
FIGURE 12 HUGGINGFACE LOGO	29
FIGURE 13 WORK BREAKDOWN STRUCTURE	31
FIGURE 14 COMPANY DIRECT COSTS 1	32
FIGURE 15 COMPANY DIRECT COSTS 2	32
FIGURE 16 COMPANY DIRECT COSTS 3	32
FIGURE 17 COMPANY INDIRECT COSTS	33
FIGURE 18 COMPANY DEVICES & LICENSES 1.....	33
FIGURE 19COMPANY DEVICES & LICENSES 2	33
FIGURE 20 SUMMARY OF COMPANY'S PROFITABILITY	33
FIGURE 21 COSTS BUDGET RESEARCH AND FOLLOWING ITEM 1.....	34
FIGURE 22 COSTS BUDGET RESEARCH AND FOLLOWING ITEM 2.....	35
FIGURE 23 COSTS BUDGET DEVELOPMENT ITEM 1	36
FIGURE 24 COSTS BUDGET DEVELOPMENT ITEM 2	38
FIGURE 25 COSTS BUDGET DOCUMENTATION ITEM 1	38
FIGURE 26 COSTS BUDGET DOCUMENTATION ITEM 2	39
FIGURE 27 COSTS BUDGET SUMMARY	39
FIGURE 28 CLIENT BUDGET.....	40
FIGURE 29 CLIENT BUDGET OVERVIEW.....	40
FIGURE 30 SYSTEM USE CASES	45
FIGURE 31 DETECT INAPPROPRIATE MESSAGES USE CASE	45
FIGURE 32 CHANGE CLASSIFICATION USE CASE.....	45
FIGURE 33 LOG IN AS ADMINISTRATOR USE CASE	45
FIGURE 34 TRAIN MODELS USE CASE	46
FIGURE 35 CORRECT PREDICTIONS USE CASE	46
FIGURE 36 SAVE RESULTS TO FILE USE CASE.....	46
FIGURE 37 SUBSYSTEM DIAGRAM.....	46
FIGURE 38 CLASS DIAGRAM IN ANALYSIS PHASE.....	48
FIGURE 39 CLASSIFICATIONMODULE CLASS	48
FIGURE 40 AUTHENTICATIONMODULE CLASS	49
FIGURE 41 BINARYCLASSIFIER CLASS	49
FIGURE 42 MULTIITEMCLASSIFIER CLASS.....	49
FIGURE 43 BINARYPREDICTION CLASS	50

FIGURE 44 MULTITEMPREDICTION CLASS	50
FIGURE 45 CHANGE CLASSIFICATION METHOD USE CASE AND SCENARIOS	50
FIGURE 46 DETECT INAPPROPRIATE MESSAGES USE CASE AND SCENARIOS	51
FIGURE 47 SAVE RESULTS TO A FILE USE CASE AND SCENARIOS.....	51
FIGURE 48 CORRECT PREDICTIONS USE CASE AND SCENARIOS	52
FIGURE 49 LOG IN AS ADMINISTRATOR USE CASE AND SCENARIOS	53
FIGURE 50 TRAIN MODELS USE CASE AND SCENARIOS.....	53
FIGURE 51 RELATION BETWEEN SCENARIOS AND USE CASES	54
FIGURE 52 MAIN WINDOW.....	55
FIGURE 53 MAIN WINDOW, START OF APPLICATION	55
FIGURE 54 MAIN WINDOW, MESSAGE TYPED IN AREA, CLASSIFICATION NOT YET PERFORMED, NON-ADMINISTRATOR USER	56
FIGURE 55 MAIN WINDOW, MESSAGE UPLOADED BY FILE, CLASSIFICATION NOT YET PERFORMED, NON-ADMINISTRATOR USER	56
FIGURE 56 MAIN WINDOW, CLASSIFICATION PERFORMED, NON-ADMINISTRATOR USER	57
FIGURE 57 MAIN WINDOW, START AS ADMINISTRATOR.....	57
FIGURE 58 MAIN WINDOW, MESSAGE TYPED IN AREA, CLASSIFICATION NOT YET PERFORMED, ADMINISTRATOR	58
FIGURE 59 MAIN WINDOW, MESSAGE UPLOADED BY FILE, CLASSIFICATION NOT YET PERFORMED, ADMINISTRATOR.....	58
FIGURE 60 MAIN WINDOW, CLASSIFICATION PERFORMED, ADMINISTRATOR	59
FIGURE 61 AUTHENTICATION WINDOW	59
FIGURE 62 TRAIN MODEL WINDOW	60
FIGURE 63 NAVIGABILITY DIAGRAM.....	61
FIGURE 64 CHANGE CLASSIFICATION METHOD USE CASE TEST	62
FIGURE 65 DETECT INAPPROPRIATE MESSAGES USE CASE TEST.....	63
FIGURE 66 SAVE RESULTS TO A FILE USE CASE TEST	63
FIGURE 67 CORRECT PREDICTIONS USE CASE TEST.....	63
FIGURE 68 LOG IN AS ADMINISTRATOR USE CASE TEST	64
FIGURE 69 TRAIN MODELS USE CASE TEST.....	65
FIGURE 70 PACKAGE DIAGRAM.....	66
FIGURE 71 COMPONENT DIAGRAM.....	68
FIGURE 72 CONTROLLER PACKAGE CLASS DIAGRAM	69
FIGURE 73 AUTHENTICATION PACKAGE CLASS DIAGRAM	70
FIGURE 74 CLASSIFIERS CLASS DIAGRAM	71
FIGURE 75 DOMAIN CLASS DIAGRAM	72
FIGURE 76 USER INTERFACE CLASS DIAGRAM	73
FIGURE 77 FULL SYSTEM CLASS DIAGRAM.....	73
FIGURE 78 DETECT INAPPROPRIATE MESSAGES USE CASE INTERACTION DIAGRAM	74
FIGURE 79 TRAIN MODEL USE CASE INTERACTION DIAGRAM	75
FIGURE 80 CORRECT PREDICTION USE CASE INTERACTION DIAGRAM	76
FIGURE 81 LOG IN AS ADMINISTRATOR USE CASE INTERACTION DIAGRAM	77
FIGURE 82 DATABASE ENTITY-RELATIONSHIP DIAGRAM	79
FIGURE 83 FINAL MAIN WINDOW.....	79
FIGURE 84 FINAL AUTHENTICATION WINDOW	80
FIGURE 85 FINAL TRAINING WINDOW	80
FIGURE 86 CONFIRMATION WINDOW FOR CORRECTING A PREDICTION	81
FIGURE 87 CONFIRMATION WINDOW FOR TRAINING A MODEL	81
FIGURE 88 CHANGE CLASSIFICATION METHOD UNIT TEST	82
FIGURE 89 DETECT INAPPROPRIATE MESSAGES UNIT TEST	83
FIGURE 90 SAVE RESULTS TO FILE UNIT TEST.....	83
FIGURE 91 CORRECT PREDICTIONS UNIT TEST	84

FIGURE 92 LOG IN AS ADMINISTRATOR UNIT TEST	85
FIGURE 93 TRAIN MODELS UNIT TEST	85
FIGURE 94 USER PROFILE RANKING SURVEY	86
FIGURE 95 GUIDED ACTIVITIES SURVEY	87
FIGURE 96 QUICK QUESTIONS ABOUT THE APPLICATION SURVEY.....	87
FIGURE 97 TESTER SURVEY	88
FIGURE 98 PERFORMANCE TESTS.....	89
FIGURE 99 VISUAL STUDIO CODE LOGO.....	91
FIGURE 100 GIT LOGO	92
FIGURE 101 NATURAL LANGUAGE PROCESSING GRAPH.....	93
FIGURE 102 CONFUSION MATRIX.....	100
FIGURE 103 DOG/CAT CONFUSION MATRIX.....	101
FIGURE 104 STATISTICS FROM STATE-OF-ART BINARY MODELS.....	104
FIGURE 105 STATISTICS FROM STATE-OF-ART MULTILABEL MODELS.....	105
FIGURE 106 BINARY MODEL ALTERNATIVES	108
FIGURE 107 ACCURACY GROUPED BY DATA TRANSFORMATION METHODS	110
FIGURE 108 ACCURACY GROUPED BYU PREPROCESSING METHODS	111
FIGURE 109 ACCURACY GROUPED BY DATASET	111
FIGURE 110 PRECISION GROUPED BY DATASET	112
FIGURE 111 MULTILABEL MODEL ALTERNATIVES.....	114
FIGURE 112 LABELS' METRIC VALUES FOR MODEL 4 WITH BAG OF WORDS	115
FIGURE 113 LABELS' METRIC VALUES FOR MODEL 4 WITH TF-IDF.....	116
FIGURE 114 LABELS' METRIC VALUES FOR MODEL 5 WITH BAG OF WORDS	116
FIGURE 115 LABELS' METRIC VALUES FOR MODEL 5 WITH TF-IDF.....	116
FIGURE 116 CHANGE CLASSIFICATION METHOD UNIT TEST RESULT	118
FIGURE 117 DETECT INAPPROPRIATE MESSAGES UNIT TEST RESULT	119
FIGURE 118SAVE RESULTS TO FILE UNIT TEST RESULT	119
FIGURE 119 CORRECT PREDICTIONS UNIT TEST RESULT	120
FIGURE 120 LOG IN AS ADMINISTRATOR UNIT TEST RESULT	121
FIGURE 121 TRAIN MODELS UNIT TEST RESULT	121
FIGURE 122 USER PROFILE RANKING SURVEY COMPLETED BY JORGE ANTONIO	122
FIGURE 123 USER PROFILE RANKING SURVEY COMPLETED BY MARTA	123
FIGURE 124 USER PROFILE RANKING SURVEY COMPLETED BY JOSE IGNACIO	123
FIGURE 125 GUIDED ACTIVITIES SURVEY COMPLETED BY JORGE ANTONIO	124
FIGURE 126 GUIDED ACTIVITIES SURVEY COMPLETED BY MARTA	125
FIGURE 127 GUIDED ACTIVITIES SURVEY COMPLETED BY JOSE IGNACIO	125
FIGURE 128 QUICK QUESTION ABOUT THE APPLICATION SURVEY COMPLETED BY JORGE ANTONIO.....	126
FIGURE 129 QUICK QUESTION ABOUT THE APPLICATION SURVEY COMPLETED BY MARTA	127
FIGURE 130 QUICK QUESTION ABOUT THE APPLICATION SURVEY COMPLETED BY JOSE IGNACIO	128
FIGURE 131 CUSTOMIZED ACCESSIBILITY CHECKLIST.....	130
FIGURE 132 PERFORMANCE TESTS RESULTS	131
FIGURE 133 INSTALLATION MANUAL OPEN CMD	132
FIGURE 134 INSTALLATION MANUAL VIRTUAL ENVIRONMENT ACTIVATION	133
FIGURE 135 EXECUTION MANUAL INIT.JSON	133
FIGURE 136 START OF APPLICATION	134
FIGURE 137 STEP 1 TO PREDICT A MESSAGE.....	135
FIGURE 138 FINAL STEP TO PREDICT A MESSAGE	135
FIGURE 139 STEP 1 TO PREDICT MESSAGES IN A FILE	136
FIGURE 140 STEP 2 TO PREDICT MESSAGES IN A FILE	136
FIGURE 141 FINAL STEP TO PREDICT MESSAGES IN A FILE.....	137

FIGURE 142 CHANGE THE METHOD FOR CLASSIFICATION.....	138
FIGURE 143 EXAMPLE OF PREDICTING WITH THE ITEMIZED METHOD	138
FIGURE 144 FILE EXPLORER FOR SAVING RESULTS TO A FILE	139
FIGURE 145 EXAMPLE OF SAVING PREDICTION TO FILE.....	139
FIGURE 146 EXAMPLE OF SAVED PREDICTION .CSV FILE.....	140
FIGURE 147 EXAMPLE OF SAVED PREDICTION .TXT FILE.....	140
FIGURE 148 AUTHENTICATION DIALOG.....	140
FIGURE 149 MAIN WINDOW AFTER LOGGING AS ADMINISTRATOR	141
FIGURE 150 PREDICTION PERFORMED AS ADMINISTRATOR.....	141
FIGURE 151 CHOSEN MESSAGE TO CORRECT PREDICTION	142
FIGURE 152 CONFIRMATION WINDOW FOR CORRECTION OF A PREDICTION	142
FIGURE 153 MESSAGE AFTER CORRECTING THE PREDICTION	143
FIGURE 154 TRAIN MODEL WINDOW	143
FIGURE 155 CONFIRMATION WINDOW FOR TRAINING A MODEL	144
FIGURE 156 CONFIRMATION WINDOW DURING TRAINING	144
FIGURE 157 FINALIZED TRAINING WITH INVALID MESSAGES.....	144
FIGURE 158 FINALIZED SUCCESSFUL TRAINING.....	145
FIGURE 159 FINAL WORK BREAKDOWN STRUCTURE	152
FIGURE 160 FINAL COSTS BUDGET RESEARCH AND FOLLOWING ITEM 1	152
FIGURE 161 FINAL COSTS BUDGET RESEARCH AND FOLLOWING ITEM 2	153
FIGURE 162 FINAL COSTS BUDGET DEVELOPMENT ITEM 1.....	154
FIGURE 163 FINAL COSTS BUDGET DEVELOPMENT ITEM 2.....	155
FIGURE 164 FINAL COSTS BUDGET DOCUMENTATION ITEM 1.....	156
FIGURE 165 FINAL COSTS BUDGET DOCUMENTATION ITEM 2.....	157
FIGURE 166 FINAL COSTS BUDGET SUMMARY.....	157
FIGURE 167 CONTENTS OF THE ATTACHED FILE	169
FIGURE 168 CONTENTS OF THE COMPRESSED FILE	169
FIGURE 169 MEETING MINUTES	170

Chapter 1. Project Report

The main features of this project, as well as other relevant characteristics are described below.

1.1 Motivation Summary, Objectives and Project Scope

Nowadays, the spread utilization of the Internet and more specifically of social media is undeniable. This exposition to the general public, besides the relentless success and benefits, inevitably leads to the proliferation of harming practices, both for the people using the web and for the Internet itself. Among these practices we can encounter posts that hold inadequate behaviours, which have been especially relevant for the last years, and a focal point for many studies and analysis.

The ease for expressing one's opinion in addition to the anonymity's apparent safety behind the screen, encourages a lot of people to spread negative-connotation messages, which could carry harmful intentions. There is a wide spectrum discussing negative opinions. On one hand, there are messages that, although negative, their purpose may be considered as a respectful review. On the other hand, there are publications whose eventual objective is hate speech, threatening, harassment, etc.

The later example has driven to an infinity of debates about the boundaries of free speech. One answer shared by several of the most visited websites has been the implementation of filtering systems, that allow taking decisions about these sorts of posts. Some companies tend to display warnings to the people that may be concerned, whilst other corporations choose to erase the messages and even punish the responsible user.

While it is true that these mechanisms may be interpreted as a limitation of freedom of speech, the organizations and their policies are the ones that establish how the filtering methods should be used, whether they ought to be used with a strong or weak approach.

Thus, inappropriate message detection systems do not present any sort of ethical dilemma in terms of its use, and they are increasingly necessary in the exponential usage of social networks and websites.

The main goal of this project is to offer whoever person or whichever company the possibility of tracking the possible harmful comments inside a set of documents. The system also has the objective of being maintainable, given that a company using it may have an already established machine learning model that can perform better classification than the ones suggested, or the company may want to create a new model from scratch. This project has not the intention of providing censorship mechanisms to the companies utilizing it, although this issue is out of the system's reach.

1.2 Summary of all Aspects

On this section, the objectives of later parts of the document are encapsulated.

Introduction. Project precedents are developed, as well as their justification, goals, and current situation.

Theoretical aspects. Brief explanation of the most abstract concepts upon which this project is supported, their importance within it and their origin.

Project planning and initial budgets. The project framework is shown on this chapter, the context in which is proposed and first outlined budgets.

Analysis. Requirements and documentation corresponding to the analysis phase within the project fulfilment.

System design. The disposition of the different parts of the system are laid out, as it can be its infrastructure, communication between subsystems or the description and sketches of the user interface, in addition to the detailed process of the testing plan.

System implementation. On this section, the document explains in detail the different tools and technologies used, along a journey through the process of the system implementation. Also, problems found during the development are specified.

Evaluation of alternatives. Discussion and analysis of the state-of-art solutions regarding this project's matter. Also, the different alternatives attempted throughout the system development are described and evaluated.

Testing development. The development of the testing plans already discussed on the system design.

System manuals. Explanations and tutorials on different usage levels, like installation manuals or user guide.

Conclusions and future work. Conclusions from the author regarding the obtained results and his expectations, and possible extensions for the system.

Project planning and final budgets. Idem to the namesake chapter, but on the final phase of this project.

Bibliographic references. Consulted resources for the project development.

Annexes. They contain the glossary of terms, a description of the attached content, and meeting minutes.

Chapter 2. Introduction

In this chapter the main discussions topics are the justification of this project, its main objectives, and the analysis of the current state-of-art solutions.

2.1 Justification of the Project

This project is born out of need of different organizations exposed to the public of locating those messages that are inappropriate. The term inappropriate is baffling, given that it varies depending on the context of its use. In this project we can define it as a property socially conferred to whichever message that exhibits elements of toxicity, obscenity, threat, insult and/or hate speech within its meaning, i.e., a message is inappropriate when most of the people analysing it recognizes that its nature is contained inside one or more of the former categories.

A large quantity of inappropriate messages in, for instance, a social network, translates to a risk for the rest of the network, since the publications are potentially detrimental for readers who may feel identified with them.

At first glance, the implementation of a system which detects these messages may seem like a controversial measure, because it can drive to a restriction to free speech, besides the fact the line drawn for texts considered attacks to an individual or group of individuals' sensitivity is vague and subjective. Although this is an unsolved debate, the discussed system is external to it, given that it is, indeed, the organization who is in possession of the system the one that decides which strategy to stick to in order to deal with these messages, either by censorship or less extreme means.

Nevertheless, there are messages that have no room in present society and should not be shared across any platform, like incitement to hatred against certain groups or threats to certain individuals.

Unfortunately, there have been multiple cases of cyberbullying that have led to dreadful events due to the widespread growth of messages with degrading intentions. However, we are currently giving more importance to mental health and commencing to talk about this topic, that not so long ago was ignored.

It is pending assignment of society to try to eradicate any sort of offensive attitude towards people or groups with the mere purpose of harming, and this system intends to be helpful on this matter.

2.2 Objectives of the Project

Next, the goals of the fulfilment of this project are listed:

1. Create a program capable of detecting a message or group of messages as inappropriate through two distinct methods:
 - a. Binary classification (appropriate or inappropriate).
 - b. A more detailed classification that breaks down the constitution of the message in terms of toxicity, obscenity, threat, insult or hate speech.
2. Allow the user to change the classification method among the options listed previously.
3. Allow an administrator (with the corresponding credentials) to train the classifiers with the intention of improving them and testing them later.
4. Allow an administrator to correct predictions performed by the both models.
5. Allow a user to save the predictions performed into a file.

2.3 Study of Current Situation

One of the features of this project, which is quite normal in text classification tasks, is the concision of the parameters by which the classification is run. This concern highlights a doubt about when a text is considered appropriate for the concept upon which it is classified (if it contains rude words, swear words, if it attacks a person or group of people despite using a correct vocabulary, etc.)

This dim definition provokes the disagreement when it comes to categorizing texts, since what a person considers inappropriate may not be inappropriate for another person. Therefore, available datasets vary drastically on the categories in which documents are classified.

Some works have focused on identifying if the used language in the text is offensive or hate speech like Davidson's, in which more classic methods are utilized, like logistic regression [1].

There are older publications like Smokey, which proposes a hostile message automatic detection system [2].

On the other hand, several other papers have done research on toxic text classification in other languages, like Banik and Rahman who cover Bengali [3], or Álvarez Carmona with Mexican Spanish [4].

In most of the cases, the model which is used is an artificial neural network trained with different datasets. Building a neural network from scratch requires a vast knowledge on deep learning and neural networks themselves. So, a common solution is using pre-trained models for the classification, like BERT [5].

The contribution of this project is the intent on proposing a definition of «inappropriate message» that gathers the former studies. It is also appealing to unify in a single system the capability of classifying a text with a binary approach or a more itemized approach with multilabel classification. Last, one of the main goals of this project is to offer a system that can be extended to receive other modules and be maintained with ease.

2.3.1 Alternatives evaluation

System alternatives can be spanned according to the sophistication degree we want to agree on. Even so, the following subsections describe some of the alternatives regarding the main model for classifying text.

2.3.1.1 Rule-based system (unfeasible)

A traditional program could be considered among the possible alternatives for implementing the system. This idea, however, is easily dispensable, due to the infinite generative nature of language, that allows building sentences in countless ways. We could cover a large sample of these sentences, but we would always fall short to the possibilities of constructing a phrase.

2.3.1.2 Logistic regression

Given that a part of our system has the goal of classifying text as inappropriate or not, we are facing a binary classification task. To solve these sorts of problems, it is frequent the use of linear classifiers. In this group of classifiers, logistic regression is one of the most common. Logistic regression allows us to make a classification with ease. There are other linear classifiers that may serve for the implementation of the system, but this is the one chosen for our task [6].

2.3.1.3 Other binary models

When it comes to selecting a classifier, the research throughout the years shows that it does not exist a perfect fit for a given task. Therefore, we must try several options in order to spot the most suitable one. To achieve a binary classification, we may look up for different sorts of models. Popular classifiers include the perceptron, stochastic gradient descent and naive bayes, among others [7].

2.3.1.4 Neural networks

Deep learning is one of the usual go-to options in artificial intelligence problems, and this system is not an exception. The capability of neural networks of retracing the error across their neurons, is an ideal way of fine-tuning a model to accomplish our task. However, knowledge about this model is not trivial. One of the main drawbacks of artificial networks is its high-demanding understanding on how it operates, and for that reason it is seen sometimes as an excessive approach depending on the task reach [8].

2.3.1.5 One vs Rest approach

This method transforms a multiclass problem into several binary tasks. This mechanism enables to use a simple and known binary model like the one we use for the binary classification and train it with a non-binary dataset. Nevertheless, it is a strategy used to «patch» in some way a more complex problem, so it probably will not be as effective as a neural network [9]. Also, it is not the best approach for multilabel tasks.

2.3.1.6 Multilabel task transformations

Multilabel strategy is a complex task that can be very-well implemented using distinct methods, like binary relevance, classifier chain, and label powerset. These methods will be described in section 2.3.1.6.1, 2.3.1.6.2, and 2.3.1.6.3. All the examples have been acquired from [11]. They are similar approaches to the OvR technique, in the sense that they try to transform the problem into a binary classification problem, and this may lead to the same issues [10].

2.3.1.6.1 Binary relevance

This method allows to split a multilabel task into multiple binary classification tasks. Imagine we had this small dataset. X1, X2 and X3 are the samples and they had three associated prediction values.

X	Target1	Target2	Target3
X1	0	0	1
X2	0	0	0
X3	1	0	1

Figure 1 Multilabel classification examples

If we performed binary relevance, we would get three datasets corresponding to each target. Thus, we have shifted a single multilabel classification problem into three binary classification problems.

X	Target1
X1	0
X2	0
X3	1

X	Target2
X1	0
X2	0
X3	0

X	Target3
X1	1
X2	0
X3	1

Figure 2 Binary relevance transformation example

This method works similarly to the One VS Rest method, but it manages non-exclusive targets.

2.3.1.6.2 Classifier chain

One of the main disadvantages of the binary relevance transformation is that it loses all correlation between labels, which depending on the task it would be interesting to keep. The classifier chain approach solves this issue adding as a feature the previous target value sequentially. Consider the example on Figure 1. If we used the classifier chain mechanism, we would first obtain a dataset with the three samples and the three values of Target1. Then, this dataset is used as the input for the next one that will have as Target2 as the target. Finally, this second dataset will be the features for the final dataset that will have Target3 as target.

X	Target1
X1	0
X2	0
X3	1

→

X	Y1	Target2
X1	0	0
X2	0	0
X3	1	0

→

X	Y1	Y2	Target3
X1	0	0	1
X2	0	0	0
X3	1	0	1

Figure 3 Classifier chain transformation example

2.3.1.6.3 Label powerset

This method allows to consider every possible combination of the targets as a singular value. The difference is that this method transforms the multilabel task into a multiclass task. We will

see how it would look in the example given in Figure 1. In this case, as we only have three of all the possible combinations, we will have three classes out of the possible eight.

X	Class
X1	1
X2	2
X3	3

Figure 4 Label powerset example

This last method could work for small number of targets but not for large numbers of targets. Some multilabel classification tasks work with hundreds of targets, so if for instance we had a complete dataset with most of the targets' possible combinations, the label powerset would compute more than 2^{100} classes, which is computationally inefficient.

Chapter 3. Theoretical Aspects

This chapter details some essential characteristics for the purpose of understanding the logic behind several of the key concepts of this project.

3.1 Artificial intelligence (AI)

The next three subparts try to discern the concepts of artificial intelligence, machine learning and deep learning, that are crucial in these sorts of problems but often confused.

The Artificial Intelligence concept has several suitable definitions, yet we will stick to the definition proposed by John McCarthy in which artificial intelligence is «the ability of a machine to mimic the behaviour of the human mind on problem-solving and decision-making» [12].

We establish the origin of artificial intelligence in the first half of the 20th century, when Alan Turing, one of the fathers of modern computer science brings up the question: «Can machines think? » on his study Computing Machinery and Intelligence [13]. His early passing and the few resources of that time prevented him from putting into practice his question, but he set the theoretical framework and the basis upon which future work would be done. Since then, artificial intelligence has evolved drastically and surpassed Turing expectations, and it is considered one of most critical goals of future's computer science [14].

3.2 Machine learning (ML)

Machine learning is a branch of artificial intelligence that gathers the methods and techniques for imitating human learning behaviour. In essence, machine learning systems manage historical labelled data and attempts to predict outputs which are not programmatically specified.

We can confer the invention of the term to Arthur L. Samuel in [15], which talks about a machine he built to play against in the game of checkers. This research proposes some approaches to creating a “learning” machine, like alpha-beta pruning [16] and the minimax algorithm [17] [18].

The main distinct feature of machine learning compared to artificial intelligence is that the learning and predicting mechanisms are intrinsic to it, whilst there are fields inside artificial intelligence that may not include these mechanisms, like some parts of natural language processing.

3.3 Deep learning (DL)

Deep learning is a subset of machine learning, where developed systems try to imitate the structure of the human brain. Its foundation lies in the neural networks of the brain.

The origin of deep learning can be set on 1943, when Warren McCulloch and Walter Pitts designed a neural-network-based computer [19].

Later research would incorporate key attributes of deep learning, like back propagation, learning rate, etc.

Although it may be seen as two identical ideas, machine learning and deep learning have clear distinctions between each other, as deep learning algorithms may develop complex patterns that machine learning mechanisms would not.

3.4 Linear regression

Although is not explicitly used on the built system, understanding linear regression is a good starting point to understand the rest of theoretical concepts.

Regression is a statistical method which allows to predict a dependent value having an independent one. In supervised machine learning, that is, machine learning which works with known data, classification and regression are the two main types. Classification attempts to predict an input with a discrete value, a finite set of results. Regression tries to correlate an input with a continuous value, which means infinite possible solutions.

Linear regression is a mathematical model used in this purpose. The general formula for this model is

$$f(x) = \beta_0 + \beta_1 X$$

Figure 5 Formula of a simple linear regression

where X is the sample given as input, β_1 is the parameter corresponding to that variable and the one we want to fit, and β_0 is a particular bias. As we can see, this is the general equation for a line function, which will predict any new input. In short, we want to find a line function which represents the tendency of the data, so when new data is presented, the model finds a nearby result to the accurate one.

This function can be generalized to cover multiple independent variables. This case is called multiple linear regression, whilst the former one is named simple linear regression.

In order to rectify the possible error of the model, we need a mechanism that computes how far we have ended up from the correct answer and recalculate the parameter (weight) accordingly. We compute the cost function. A popular method is the mean square error estimation, which elevates to the square the error and reassigns new values to the parameters [20]–[22].

3.5 Logistic regression

Logistic regression is a statistical model which allows to predict a categorical value given an input. Given an independent variable or set of variables, the resolution will be a value in a binary set (0 or 1, true or false, yes or no, etc.). Despite having «regression» in its name, it is enclosed in classification tasks with discrete values. The behaviour of this model is due to the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Figure 6 First formula of logistic regression for one variable

This is a “S” shaped function that allows to clearly split two different groups and normalize the output. To achieve a result equal to 0 or 1, we need a threshold that separates both groups. This threshold usually lies around 0.5 (anything above or equal to that number would be 1 and anything below would be 0), so each possible value weights the same, however this is also a parameter we can tune for a granted problem. We could redraft the former function as:

$$f(x) = g(\beta_0 + \beta_1 X)$$

Figure 7 Second formula of logistic regression for one variable

In this case $g(x)$ would be the sigmoid function. It is unequivocal that logistic regression could be seen as a linear regression model with a certain function which permits the discretization of the result.

For the cost functions we have, once again, several options. Maximum likelihood, Newton’s method, stochastic gradient descent, etc. Selecting the optimal cost function will depend in the problem we want to solve.

As for every model, we would need a mechanism to prevent overfitting, which is occurs whenever training data is classified correctly but unseen data is prone to error. Hence, logistic regression uses penalization algorithms which enlarge the penalization to a greater or lesser extent [6], [23], [24].

3.6 Stochastic Gradient Descent

To understand Stochastic Gradient Descent, we should know what Gradient Descent is first. Gradient Descent is an iterative algorithm whose objective is to find the values for a set of independent variables, or features, that produce a minimum point inside a given function. This is considered an algorithm of optimization, a mathematical field which consists of finding the optimal solution for a set of elements. In machine learning, this function usually is referred as the cost function or loss function, which is the function we want to minimize.

As this algorithm follows an iterative process, the steps of the algorithm are explained inside an example with two features $x = (x_1, x_2)$ and one target y_1 :

1. Pick random values for the features.
2. Calculate the gradient of the function. The gradient of the function is the vector formed by the partial derivatives of the target. This would be $\nabla f = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} \end{bmatrix}$. This is a vector that represents the slope of this function.
3. Recalculate the parameters adding the opposite of this gradient ($x = x - \alpha \nabla f$), being α a hyperparameter called "learning rate". This parameter will establish how much this gradient will change the recomputed parameters. In essence, it marks the size of the step that the algorithm performs. A small learning rate could lead to excessive iterations. A large learning rate could mean no convergence. The value of this hyperparameter is key for a good gradient descent algorithm.
4. Repeat 2 and 3 until the gradient becomes near 0.

The problem comes when we work with large numbers. If we had a one-million-sample dataset, the gradient descent algorithm would have to use for completing a single iteration. Thus, it would not be resource-efficient to perform it. Stochastic Gradient Descent (SGD) solves this issue by selecting stochastically (randomly) a single sample to perform each iteration [25]–[27].

3.7 Neural network

Artificial neural networks are deep learning models which represent an abstraction of the human brain structure from the real world. The atomic structure in a neural network is a node called neuron.

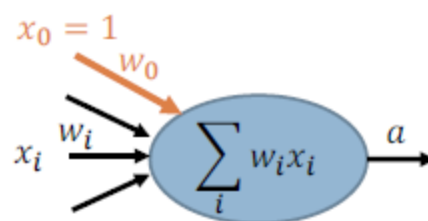


Figure 8 Representation of an artificial neuron

In this figure the most important characteristics of a neuron are presented. x_i is the i^{th} input, w_i is the weight corresponding to the i^{th} input, x_0 and w_0 are a biased input and weight, respectively, and a is the output of the operation inside the node.

One of the limitations of a single neuron is that it cannot solve non-linear problems, so they are concatenated with other neurons to form sequences, and they are also usually group into layers. A layer is a set of neurons that receive the same inputs, and it is usually represented vertically. To achieve more interesting alterations on each node the neurons apply activation functions,

which perform non-linear operations, for instance the sigmoid or the hyperbolic tangent function.

The most intriguing feature of neural networks and deep learning models is their ability of being self-aware of the error computed and trying to fix it. Backpropagation is a method that assigns to each neuron an amount of responsibility they had on the error and does this in an efficient way [8], [28].

3.8 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a deep learning model developed by Google used in natural language processing tasks. The idea behind BERT is to pre-train a model that “understands” the language and prepare it to be fine-tuned to the specific problem. It was pretrained against over 3.3 billion words from Google Books and Wikipedia and it is composed of several layers and sequences.

Towards the goal of understanding BERT, we must understand first what Transformers is. Neural networks have evolved since its origin. For instance, convolutional neural networks are specialized in spatial learning like images, and recurrent neural networks that receive sequences like text and have into account the “memory” of previous neurons or layers. In this last case nevertheless, this memory has come to be small, since in long sentences the first words would have little semantic impact on the latest ones. Transformers attempt to solve this issue, by feeding the neural network with positional encoding of the text and applying attention mechanisms [29], [30].

3.9 Python

Python is a high-level programming language developed in the decade of 1980 that has gained increasingly relevance in the last years. It was developed by Guido van Rossum, when he was working in a programming language called ABC. The developer had the clear objective of satisfying a particular condition: implement a programming language that could be taught to intelligent computer users who had no experience in programming. Van Rossum has always assumed that most of the success of implementing Python was his experience in the ABC programming language project [31].



Figure 9 Python logo

Two of the main distinctive characteristics of this language are the readability for the human and the flexibility to fit different programming paradigms. Python is widely used by artificial intelligence developers, due to the existence of a large quantity of libraries for this purpose, and moreover, for machine learning. We will describe some of these machine learning libraries in the following subchapters.

3.10 Scikit Learn

Scikit Learn is a popular machine learning library implemented in Python. It has several options of supervised and unsupervised learning. One of the key features of Scikit Learn is that it is intuitive to implement and understand with minimum machine learning concepts. Linear models, which are the focal elements of Scikit Learn used on this project, are fast, memory-cheap, and easy-understandable. However, the main concern with this library is its lack of deep learning implementations like neural networks. There were attempts of developing libraries which would use well-known libraries implementing neural networks with a friendly Scikit Learn interface, but they are outdated and not working as expected [32].



Figure 10 Scikit Learn logo

3.11 Pytorch

PyTorch is a widely used Python open-source library specialized in machine learning and deep learning. This library is popular due to its scalable structure, where you can implement a simple model at first and gradually improve its performance by adding components. It has the capability of constructing neural networks, which is the main feature used for the systems regarding Pytorch. A considerable issue with this library is the fact that it is not as elementary to use as Scikit Learn can be, since it demands higher machine learning skills than the former library [33].



Figure 11 PyTorch logo

3.12 Transformers

Transformers is a library offered by HuggingFace, a community that creates open-source systems to implement machine learning and deep learning programs. HuggingFace is a company founded by Clement Delangue and Julien Chaumond in 2016 whose goal is the exploration of artificial intelligence. Transformers is a state-of-art machine learning library which is based on PyTorch [33], TensorFlow [34] and Jax [35], and it is widely used in both deep-learning and Natural Language Processing tasks. Its importance in this project lays in that it contains different implementations of BERT models.



Figure 12 HuggingFace logo

Chapter 4. Project planning and initial budgets

The chapter presents the planning and budget suggested at the beginning of this project. Also, a simulated company model is set, mocking different job roles, salaries and costs and benefits.

4.1 Initial planning

The first draft made of this project was finally set on November of 2021, where it is established a duration of 93 days (around 3 months) that would cover mainly the first quarter of 2022, from November 22th, 2021, until March 3rd, 2022, when it is expected to be delivered.

The plan is developed by only person represented in different roles, based on which would do that specific task inside a real-world organization.

ID	Name	Start	Finish	Resource Names
1	Project	Mon 22/11/21	Wed 30/03/22	
2	Research	Mon 22/11/21	Thu 02/12/21	
3	Related papers	Mon 22/11/21	Wed 24/11/21	Team Leader
4	Implementation alternatives	Thu 25/11/21	Mon 29/11/21	Team Leader
5	Implementation tools	Tue 30/11/21	Thu 02/12/21	Team Leader
6	Development	Fri 03/12/21	Fri 18/03/22	
7	System 1 Alternative	Fri 03/12/21	Mon 28/02/22	
8	Analysis	Fri 03/12/21	Wed 08/12/21	
9	System definition	Fri 03/12/21	Mon 06/12/21	Analyst
10	Elicitation	Tue 07/12/21	Wed 08/12/21	Analyst
11	Design	Thu 09/12/21	Thu 16/12/21	
12	Architecture design	Thu 09/12/21	Mon 13/12/21	Software Engineer
13	Diagrams and models design	Tue 14/12/21	Thu 16/12/21	Software Engineer
14	Development	Fri 17/12/21	Tue 08/02/22	
15	Implementation	Fri 17/12/21	Thu 30/12/21	
16	NLP	Fri 17/12/21	Thu 23/12/21	Senior Programmer
17	Machine learning	Fri 24/12/21	Thu 30/12/21	Senior Programmer
18	Training	Fri 31/12/21	Wed 19/01/22	Senior Programmer
19	Validation	Thu 20/01/22	Tue 08/02/22	Senior Programmer
20	Testing	Wed 09/02/22	Mon 28/02/22	
21	Unit testing	Wed 09/02/22	Fri 18/02/22	Tester
22	Acceptance testing	Mon 21/02/22	Mon 28/02/22	Tester
23	System 2 Alternative	Fri 03/12/21	Mon 28/02/22	
24	Analysis	Fri 03/12/21	Wed 08/12/21	
25	System definition	Fri 03/12/21	Mon 06/12/21	Analyst
26	Elicitation	Tue 07/12/21	Wed 08/12/21	Analyst
27	Design	Thu 09/12/21	Thu 16/12/21	
28	Architecture design	Thu 09/12/21	Mon 13/12/21	Software Engineer
29	Diagrams and models design	Tue 14/12/21	Thu 16/12/21	Software Engineer
30	Development	Fri 17/12/21	Tue 08/02/22	
31	Implementation	Fri 17/12/21	Thu 30/12/21	

32	NLP	Fri 17/12/21	Thu 23/12/21	Senior Programmer
33	Machine learning	Fri 24/12/21	Thu 30/12/21	Senior Programmer
34	Training	Fri 31/12/21	Wed 19/01/22	Senior Programmer
35	Validation	Thu 20/01/22	Tue 08/02/22	Senior Programmer
36	Testing	Wed 09/02/22	Mon 28/02/22	
37	Unit testing	Wed 09/02/22	Fri 18/02/22	Tester
38	Acceptance testing	Mon 21/02/22	Mon 28/02/22	Tester
39	Authentication	Tue 01/03/22	Fri 04/03/22	Senior Programmer
40	Interface	Mon 07/03/22	Fri 18/03/22	Senior Programmer
41	Documentation	Fri 03/12/21	Wed 30/03/22	
42	Planning and budget	Fri 03/12/21	Mon 06/12/21	Team Leader
43	Report and introduction	Tue 07/12/21	Tue 07/12/21	Team Leader
44	Theoretical aspects	Tue 07/12/21	Wed 08/12/21	Team Leader
45	Analysis	Thu 09/12/21	Mon 13/12/21	Analyst
46	Design	Fri 17/12/21	Tue 21/12/21	Software Engineer
47	Implementation	Mon 21/03/22	Wed 23/03/22	Software Engineer
48	Testing	Tue 01/03/22	Wed 02/03/22	Tester
49	Annexes	Thu 24/03/22	Fri 25/03/22	Software Engineer
50	Conclusions	Mon 28/03/22	Wed 30/03/22	Team Leader
51	Follow-up meetings	Thu 25/11/21	Thu 13/01/22	
52	Follow-up meeting 1	Thu 25/11/21	Thu 25/11/21	Project Leader; Team Leader
53	Follow-up meeting 2	Thu 02/12/21	Thu 02/12/21	Project Leader; Team Leader
54	Follow-up meeting 3	Thu 09/12/21	Thu 09/12/21	Project Leader; Team Leader
55	Follow-up meeting 4	Thu 16/12/21	Thu 16/12/21	Project Leader; Team Leader
56	Follow-up meeting 5	Thu 23/12/21	Thu 23/12/21	Project Leader; Team Leader
57	Follow-up meeting 6	Thu 30/12/21	Thu 30/12/21	Project Leader; Team Leader
58	Follow-up meeting 7	Thu 06/01/22	Thu 06/01/22	Project Leader; Team Leader
59	Follow-up meeting 8	Thu 13/01/22	Thu 13/01/22	Project Leader; Team Leader

Figure 13 Work Breakdown Structure

4.2 Initial budget

To explain the budget suggested at the beginning of the implementation of this project, it has been established into a made-up company with different roles, that simulate the different parts of work of done by the author and director, and salaries which will be discussed in the following subchapter, in addition the indirect costs and amortizations. Later, the budget itself is presented, both for the costs and client.

4.2.1 Company definition

The company that is responsible of the fulfilment of the development of the system required consists of six roles: Project Leader, Team Leader, Software Engineer, Analyst, Senior Programmer and Tester. In Figure 14, Figure 15, and Figure 16 salaries, productivity, annual worked hours, price per hour of each role and total invoice is shown.

Staff	Quantity	Gross salary/year (€)	Salary cost/year (€)	Total (€)
Project Leader	1	45,000.00	59,850.00	59,850.00
Senior Programmer	1	22,000.00	29,260.00	29,260.00

Software Engineer	1	28,000.00	37,240.00	37,240.00
Tester	1	22,000.00	29,260.00	29,260.00
Analyst	1	25,000.00	33,250.00	33,250.00
Team Leader	1	30,000.00	39,900.00	39,900.00
Total	6			228,760.00

Figure 14 Company Direct Costs 1

Prod. (%)	Direct cost (€)	IC (%)	Indirect Cost (€)	Hours/year
20.00%	11,970.00	80.00%	47,880.00	2080
95.00%	27,797.00	5.00%	1,463.00	2080
95.00%	35,378.00	5.00%	1,862.00	2080
95.00%	27,797.00	5.00%	1,463.00	2080
90.00%	29,925.00	10.00%	3,325.00	2080
75.00%	29,925.00	25.00%	9,975.00	2080
	162,792.00		65,968.00	

Figure 15 Company Direct Costs 2

Prod. hours/year	Prod. Hours/year (total)	Price/hour (€)	Billing (€)	Price/hour (w/o benefits) (€)
416.00	416.00	80.00	33,280.00	64.00
1,976.00	1,976.00	27.50	54,340.00	22.00
1,976.00	1,976.00	35.00	69,160.00	28.00
1,976.00	1,976.00	27.50	54,340.00	22.00
1,872.00	1,872.00	33.00	61,776.00	26.50
1,560.00	1,560.00	47.50	74,100.00	38.00
	9,776.00		346,996.00	

Figure 16 Company Direct Costs 3

Note that the column Price/hour was decided to be calculated as $\frac{\text{salary cost}}{\text{productive hours/year}} * 1,85$, and for unfeasible values.

Next, we can see indirect costs of the company in Figure 17, summarizing facilities maintenance and transporting and communication expenses.

Indirect costs	Monthly Cost (€)	Annual Cost (€)
Office cleaning	1,500.00	18,000.00
Electricity consumption	200.00	2,400.00
Water consumption	50.00	600.00
Transport expenses	200.00	2,400.00
Office renting	1,000.00	12,000.00
Communication expenses	600.00	7,200.00

Office equipment	100.00	1,200.00
Audits costs	125.00	1,500.00
Sanitizing and disinfection equipment	40.00	480.00
Total		45,780.00

Figure 17 Company Indirect Costs

Afterwards, in Figure 18 and Figure 19 we may observe the material cost, amortizations, and utilization period.

Device/License	Units	Price (€)	Maintenance (€)	Total price (€)
Microsoft 365 License	1	20.50		20.50
Adobe Illustrator License	1	30.00		30.00
Windows 10 Pro License	1	5.00		5.00
Microsoft Project License	1	35.00		35.00
Development laptops	1	1,100.00	100.00	1,200.00
Office desktop PCs	1	1,200.00	200.00	1,400.00

Figure 18 Company Devices & Licenses 1

Total Cost (€)	Annual Cost (€)	Type	Time window	Amortization (3 months)
20.50	246.00	Rent	Monthly	
30.00	360.00	Rent	Monthly	
5.00	1.00	Amortization	5	0.30
35.00	7.00	Amortization	5	2.10
1,200.00	300.00	Amortization	4	68.75
1,400.00	350.00	Amortization	4	75.00
TOTAL	1,264.00			

Figure 19 Company Devices & Licenses 2

Finally, a summary of the profitability of the company is presented in Figure 20. In this section we may observe that the expected benefits represent a 25% of the total cost of the company.

Direct costs	162.792,00 €
Indirect costs	113.012,00 €
Total costs	275.804,00 €
Target benefits (25%)	68.951,00 €
Billing needs	344.755,00 €
Actual billing based on productive hours	346.996,00 €
Margin between total costs and billing	0.65%

Figure 20 Summary of Company's profitability

4.2.2 Costs budget

This section describes how the costs budget has been planned in the beginning of this project. It has been broken down to three separate items: research and following up, development, and documentation.

- **Research and following up.** It encapsulates the investigation of the basis upon which the system will be supported, searching for state-of-art solutions and current proposed papers, evaluating alternatives, and studying the basics of the corresponding theoretical aspects. In addition to this, the following-up meetings where the progress is shown are specified. See Figure 21 and Figure 22.
- **Development.** It summarises the two main systems which will be developed throughout this project. These alternatives are broken down into four phases: analysis, design, implementation, and testing. Moreover, the authentication subsystem and the user interface are detailed. See Figure 23 and Figure 24.
- **Documentation.** The elaboration of the different parts of this document. See Figure 25 and Figure 26.

Each subitem cost is calculated by the salary of the role executing it. See Figure 16.

I1	I2	I3	Description	Quantity	Units
01			Research		
	001		Related papers		
		01	Team Leader	24	hours
	002		Implementation alternatives		
		01	Team Leader	24	hours
	003		Implementation tools		
		01	Team Leader	24	hours
02			Following up		
	001		Follow-up meetings		
		01	Project Leader	4.5	hours
		02	Team Leader	4.5	hours

Figure 21 Costs Budget Research and following Item 1

I1	I2	I3	Price	Subtotal (3)	Subtotal (2)	Total
01						2,736.00 €
	001				912.00 €	
		01	38.00 €	912.00 €		
	002				912.00 €	
		01	38.00 €	912.00 €		
	003				912.00 €	
		01	38.00 €	912.00 €		
02						459.00 €

001				459.00 €	
	01		64.00 €	288.00 €	
	02		38.00 €	171.00 €	
				TOTAL	3,195.00 €

Figure 22 Costs Budget Research and following Item 2

I1	I2	I3	I4	Description	Quantity	Units
01				System 1 Alternative		
	001			Analysis		
		0001		System definition		
			01	Analyst	16	hours
		0002		Elicitation		
			01	Analyst	16	hours
	002			Design		
		0001		Architecture design		
			01	Software Engineer	24	hours
		0002		Diagrams and models design		
			01	Software Engineer	24	hours
	003			Development		
		0001		Implementation		
			01	Senior Programmer	80	hours
		0002		Training		
			01	Senior Programmer	112	hours
		0003		Validation		
			01	Senior Programmer	112	hours
	004			Testing		
		0001		Unit testing		
			01	Tester	64	hours
		0002		Acceptance testing		
			01	Tester	48	hours
02				System 2 Alternative		
	001			Analysis		
		0001		System definition		
			01	Analyst	16	hours
		0002		Elicitation		
			01	Analyst	16	hours
	002			Design		
		0001		Architecture design		
			01	Software Engineer	24	hours
		0002		Diagrams and models design		

		01	Software Engineer	24	hours
003			Development		
	0001		Implementation		
		01	Senior Programmer	80	hours
	0002		Training		
		01	Senior Programmer	112	hours
	0003		Validation		
		01	Senior Programmer	112	hours
004			Testing		
	0001		Unit testing		
		01	Tester	64	hours
	0002		Acceptance testing		
		01	Tester	48	hours
03			Authentication		
		01	Senior Programmer	64	hours
04			Interface		
		01	Senior Programmer	160	hours

Figure 23 Costs Budget Development Item 1

I1	I2	I3	I4	Price	Subtotal (4) (€)	Subtotal (3) (€)	Subtotal (2) (€)	Total
01								11,344.00
	001						848.00	
		0001				424.00		
			01	26.50	424.00			
		0002				424.00		
			01	26.50	424.00			
	002						1,344.00	
		0001				672.00		
			01	28.00	672.00			
		0002				672.00		
			01	28.00	672.00			
	003						6,688.00	
		0001				1,760.00		
			01	22.00	1,760.00			

		0002				2,464.00		
			01	22.00	2,464.00			
		0003				2,464.00		
			01	22.00	2,464.00			
	004						2,464.00	
		0001				1,408.00		
			01	22.00	1,408.00			
		0002				1,056.00		
			01	22.00	1,056.00			
02								11,344.00
	001						848.00	
		0001				424.00		
			01	26.50	424.00			
		0002				424.00		
			01	26.50	424.00			
	002						1,344.00	
		0001				672.00		
			01	28.00	672.00			
		0002				672.00		
			01	28.00	672.00			
	003						6,688.00	
		0001				1,760.00		
			01	22.00	1,760.00			
		0002				2,464.00		
			01	22.00	2,464.00			
		0003				2,464.00		
			01	22.00	2,464.00			
	004						2,464.00	
		0001				1,408.00		
			01	22.00	1,408.00			
		0002				1,056.00		

			01	22.00	1,056.00			
03								1,408.00
			01	22.00	1,408.00			
04								3,520.00
			01	22.00	3,520.00			
							TOTAL	27,616.00

Figure 24 Costs Budget Development Item 2

I1	I2	Description	Quantity	Units
01		Report and introduction		
	01	Team Leader	8	hours
02		Theoretical aspects		
	01	Team Leader	16	hours
03		Planniing and budget		
	01	Team Leader	16	hours
04		Analysis		
	01	Analyst	24	hours
05		Design		
	01	Software Engineer	24	hours
06		Implementation		
	01	Software Engineer	24	hours
07		Testing		
	01	Tester	16	hours
08		Conclusions		
		Project Leader	24	hours
09		Annexes		
	01	Team Leader	16	hours

Figure 25 Costs Budget Documentation Item 1

I1	I2	Price	Subtotal (2)	Total
01				304.00 €
	01	38.00 €	304.00 €	
02				608.00 €
	01	38.00 €	608.00 €	
03				608.00 €
	01	38.00 €	608.00 €	
04				636.00 €
	01	26.50 €	636.00 €	
05				672.00 €

	01	28.00 €	672.00 €	
06				672.00 €
	01	28.00 €	672.00 €	
07				352.00 €
	01	22.00 €	352.00 €	
08				1,536.00 €
		64.00 €	1,536.00 €	
09				608.00 €
	01	38.00 €	608.00 €	
			TOTAL	5,996.00 €

Figure 26 Costs Budget Documentation Item 2

Figure 27 Costs Budget summary is a summary of the costs budget.

Item	Item name	Total
01	Research and following up	3.195,00 €
02	Development	27.616,00 €
03	Documentation	5.996,00 €
	Total Cost	36.807,00 €

Figure 27 Costs Budget summary

4.2.3 Client budget

The following table shows the budget proposed to the client, summing up the most high-level but concise points on each item. We must take into account that each subtotal is computed as the price of each module bearing in mind the 25% of desired benefit, adding then the corresponding increment for research and following up expenses, that are not included in any item. This calculation does not apply for the Acquired Hardware item, which is mere purchase needed for training system, regardless of the benefit wanted.

Item	I1	I2	Item	Subtotal (2) (€)	Subtotal (1) (€)	Total
01			Development			37,048.75 €
	01		System 1 Alternative		15,218.75	
		01	Analysis	1,137.65		
		02	Design	1,803.07		
		03	Implementation	8,972.41		
		04	Testing	3,305.62		
	02		System 2 Alternative		15,218.75	
		01	Analysis	1,137.65		
		02	Design	1,803.07		

	03	Implementation	8,972.41		
	04	Testing	3,305.62		
	03	Authentication		1,888.93	
	04	User interface		4,722.32	
02		Acquired hardware			1,600.00 €
	01	2 GPUs for training/validation		1,600.00	
03		Documentation			8,044.04 €
	01	Report and introduction		407.84	
	02	Theoretical aspects		815.67	
	03	Planning and budget		815.67	
	04	Analysis		853.24	
	05	Design		901.53	
	06	Implementation		901.53	
	07	Design		472.23	
	08	Conclusions		2,060.65	
	09	Annexes		815.67	
			TOTAL CLIENT		46,692.79 €
			VAT		21%
			TOTAL CLIENT (APPLIED VAT)		56,498.28 €

Figure 28 Client Budget

This next table shows project integration within the company described capability.

Total Costs	36,807.00 €
Total Client	56,498.28 €
Profit (~25%)	19,691.28 €
Research and following up expenses (they are not included in any item)	3,195.00 €
Profit margin	22.80%

Figure 29 Client Budget Overview

Chapter 5. Analysis

This chapter covers the analysis phase of this project, elicitation, and proper documentation of the requisites.

5.1 System definition

We will comment the scope specification of the system, the system requirements, and the description of the use cases and scenarios, along some diagrams to take a visual glance of the system operation.

5.1.1 System scope specification

The system is intended to perform a classification based on whether a message or messages introduced by the user are inappropriate or not through machine learning mechanisms. The system offers two methods to achieve this classification: a binary classification that discretizes a message as toxic or not, and an itemized classification that evaluates the text in terms of toxicity, obscenity, threatening, insult and/or identity hate. In addition to this, an administrator can provide more training data to the system.

5.2 System requirements

This subsection details the actors found in the system and the obtained requirements for its implementation.

5.2.1 Elicitation

These section covers the functional and non-functional requirements acquired from the client.

5.2.1.1 Functional requirements

RFSIS.1. The program will allow the user to input a message to be classified.

RFSIS.1.1. The system must present the option for the user of introducing a message or set of messages.

RFSIS.1.1.1. It will show an area where the user can insert the input message. The written text will count as a single message.

RFSIS.1.1.2. It will show the option that allows to select the file containing the set of messages.

RFSIS.1.1.3. Both options are exclusive. If the user picks one option, it cannot pick the other one simultaneously.

RFSIS.1.1.4. It will check that the input is not empty.

- RFSIS.1.1.5.** The input will have a fixed maximum length of 500 characters.
 - RFSIS.1.1.5.1.** This value may be subject of change by the client.
- RFSIS.1.2.** The system must display the classification methods for the user choice.
 - RFSIS.1.2.1.** It will display the binary method.
 - RFSIS.1.2.2.** It will display the itemized method.
 - RFSIS.1.2.2.1.** The method will consist of six items.
 - RFSIS.1.2.2.1.1.** Toxic item.
 - RFSIS.1.2.2.1.2.** Severe toxic item.
 - RFSIS.1.2.2.1.3.** Obscene item.
 - RFSIS.1.2.2.1.4.** Threat item.
 - RFSIS.1.2.2.1.5.** Insult item.
 - RFSIS.1.2.2.1.6.** Identity hate item.
 - RFSIS.1.2.3.** It is mandatory that the user selects one of these methods. It will not permit continuing until one is selected.
- RFSIS.1.3.** The system will compute the result given the chosen method.
 - RFSIS.1.3.1.** While this calculation lasts, the user will be shown a message reporting the situation.
- RFSIS.1.4.** The system will present the result to the user.
 - RFSIS.1.4.1.** For the method in **RFSIS.1.2.1.** , it will show a message depending on the prediction.
 - RFSIS.1.4.1.1.** The message for an appropriate prediction is 'Appropriate'.
 - RFSIS.1.4.1.2.** The message for a non-appropriate prediction is 'Inappropriate'.
 - RFSIS.1.4.1.3.** The actual messages are subject to change by the client.
 - RFSIS.1.4.2.** For the method in **RFSIS.1.2.2.** , it will show the list of items with their individual prediction and the confidence percentage of each item prediction.
 - RFSIS.1.4.2.1.** For **RFSIS.1.2.2.1.1.** it will show 'Toxic' or 'Not Toxic'.
 - RFSIS.1.4.2.2.** For **RFSIS.1.2.2.1.2.** it will show 'Severe Toxic' or 'Not Severe Toxic'.
 - RFSIS.1.4.2.3.** For **RFSIS.1.2.2.1.3.** it will show 'Obscene' or 'Not Obscene'.
 - RFSIS.1.4.2.4.** For **RFSIS.1.2.2.1.4.** it will show 'Threat' or 'Not Threat'.
 - RFSIS.1.4.2.5.** For **RFSIS.1.2.2.1.5.** it will show 'Insult' or 'Not Insult'.
 - RFSIS.1.4.2.6.** For **RFSIS.1.2.2.1.6.** it will show 'Identity Hate' or 'Not Identity Hate'.
 - RFSIS.1.4.2.7.** The messages are subject to change by the client.
- RFSIS.1.5.** The system must show an option of redoing a classification with new messages.
 - RFSIS.1.5.1.** To perform another classification is mandatory to select this option.
- RFSIS.1.6.** The system will allow the user to save the results to a file.
 - RFSIS.1.6.1.** The extension established is .csv.
 - RFSIS.1.6.1.1.** It may be subject to change by the client.
 - RFSIS.1.6.2.** It will store the text and the values presented to the user previously.
 - RFSIS.1.6.2.1.** If the user introduced a set of messages, the output file will contain one classification in each line.
- RFSIS.2.** The system will permit a user to log in as administrator.
 - RFSIS.2.1.** The program must present an option in which the user can introduce the credentials.
 - RFSIS.2.1.1.** The program will ask for the username.
 - RFSIS.2.1.1.1.** The username can only contain alphanumeric characters.

- RFSIS.2.1.2.** The program will ask for the password.
- RFSIS.2.2.** The program must check the validity of the input data.
- RFSIS.2.2.1.** The data cannot be blank.
- RFSIS.2.2.2.** The data will be compared to a database, verifying it matches any information inside it.
- RFSIS.2.2.3.** If any validation fails, the program must warn the user.
- RFSIS.2.2.3.1.** The warning message will be 'Invalid username or password'.
- RFSIS.2.2.4.** The message is subject to change by the client.
- RFSIS.2.2.5.** If the credentials are valid, the program will return the user to the previous options.
- RFSIS.2.3.** The program must forbid the user to run as administrator if **RFSIS.2.2.** is not accomplished.
- RFSIS.3.** The system will allow the user to train the models with new data.
- RFSIS.3.1.** The user must be logged in as administrator.
- RFSIS.3.2.** The user will be presented both methods offered which are described in **RFSIS.1.2.1.** and **RFSIS.1.2.2.** to be trained.
- RFSIS.3.3.** The user will be displayed an option submitting a file with the new data.
- RFSIS.3.3.1.** The file must be in .csv format.
- RFSIS.3.3.1.1.** This may be subject to change by the client.
- RFSIS.3.3.2.** Each line of the file will contain a piece of data.
- RFSIS.3.3.2.1.** The piece of data must contain a text comment.
- RFSIS.3.3.2.1.1.** The program will check it is not blank.
- RFSIS.3.3.2.2.** The piece of data will contain the classification granted for the text.
- RFSIS.3.3.2.2.1.** This classification cannot be blank.
- RFSIS.3.3.2.2.2.** For method in **RFSIS.1.2.1.** , it will contain a number describing the possible values.
- RFSIS.3.3.2.2.2.1.** For appropriate classification it will be 0.
- RFSIS.3.3.2.2.2.2.** For inappropriate classification it will be 1.
- RFSIS.3.3.2.2.3.** For method in **RFSIS.1.2.2.** , it will contain a number for each item.
- RFSIS.3.3.2.2.3.1.** For each item it will be 0 if it is not satisfied.
- RFSIS.3.3.2.2.3.2.** For each item it will be 1 if it is satisfied.
- RFSIS.3.3.2.2.4.** This classification cannot be any other value than the described above.
- RFSIS.3.3.2.3.** Each section of the piece of data will be separated by a semicolon.
- RFSIS.3.4.** If the file is valid, the data will be trained against the corresponding model.
- RFSIS.3.5.** The program will report the user the results of the training.
- RFSIS.3.5.1.** The message for a successful training will be 'Data correctly integrated'.
- RFSIS.3.5.1.1.** This message is subject to change by the client.
- RFSIS.3.5.2.** The message for an unsuccessful training will be 'Data incorrectly integrated' in addition to the source of the error.
- RFSIS.3.5.2.1.** This message is subject to change by the client.
- RFSIS.4.** The system will allow the user to correct the results of a prediction.
- RFSIS.4.1.** The user must be logged in as administrator.
- RFSIS.4.2.** The user must set the program in the state described in **RFSIS.1.4.** .

RFSIS.4.3. The program will permit the user to introduce the desired prediction.

RFSIS.4.3.1. For the method in **RFSIS.1.2.1.** , it will show the two possible alternatives to be chosen.

RFSIS.4.3.1.1. The possible options for the correction are the ones described in **RFSIS.1.4.1.1.** and **RFSIS.1.4.1.2.** .

RFSIS.4.3.2. For the method in **RFSIS.1.2.2.** , it will display options for each item.

RFSIS.4.3.2.1. The possible options for the corrections are the ones described in the subitems of **RFSIS.1.4.2.** .

RFSIS.4.3.3. The information introduced cannot be blank.

RFSIS.4.3.4. The information introduced cannot have any other value than the ones described.

RFSIS.4.4. The program will create a file and train the corresponding model with the process described in **RFSIS.3.**

5.2.1.2 Non-functional requirements

RNFSIS.1. The system implementation must attach to well-known design pattern as much as possible.

RNFSIS.2. The password input for the administrator process described in must be encrypted before comparing it to the database.

RNFSIS.3. Execution time for prediction must be lower than 1 second per message.

RNFSIS.4. Execution time for every new cycle of training of the binary model must be lower than 1 minute.

RNFSIS.5. Execution time for every new cycle of training of the itemized model must be lower than 1 minute.

RNFSIS.6. The system must accomplish the features listed in section 9.3.

5.2.2 System Actors Identification

The system recognizes two different actors able to use it.

5.2.2.1 Non-administrator user

It is any user that is not logged in as an administrator. It can run classifications and save the results to a file. It can also try to log in as administrator if it has the needed credentials.

5.2.2.2 Logged-in user (administrator)

It has the complete utilization of the system. He may perform any operation doable by a non-administrator user. In addition to this, he can supply the models with new training data. It can also correct the prediction of one classification.

5.2.3 Use cases specification

The next diagram shows the possible use cases offered by the system.

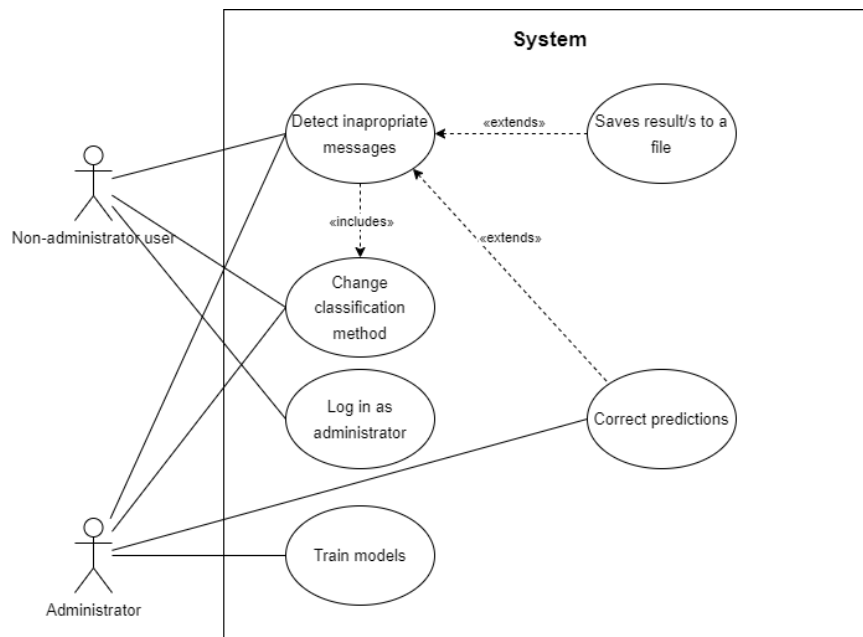


Figure 30 System use cases

Use case name
Detect inappropriate messages
Description
The user may make a classification introducing a message or set of messages. The system computes the result and shows it to the user. Moreover, the user may save the results to a file. If the user is logged as an administrator, it may correct the predictions.

Figure 31 Detect inappropriate messages use case

Use case name
Change classification method
Description
The user selects the classification method between the methods offered. This step is mandatory in order to perform the Detect inappropriate messages use case.

Figure 32 Change classification use case

Use case name
Log in as administrator
Description
Only available for users not logged in as administrator. The user uses credentials to log in as an administrator and have access to other features of the system.

Figure 33 Log in as administrator use case

Use case name

Train models
Description
It is mandatory that the user is logged in. The administrator can input data that is validated by the program and trained against the model if the validation is successful. The system warns the administrator of the result of the operation.

Figure 34 Train models use case

Use case name
Correct a prediction
Description
It is mandatory that the user is logged as an administrator. The administrator must make a classification first. Then he will have the option of selecting the new prediction values and introduce them to retrain the model with these prediction values.

Figure 35 Correct predictions use case

Use case name
Save results to file
Description
It is mandatory that at least one classification has had been performed. The user will have the option of obtaining the results in a external file.

Figure 36 Save results to file use case

5.3 Subsystems identification in analysis phase

Although this system may be thought as a seldom monolithic system, we could depict it as several subsystems interconnected.

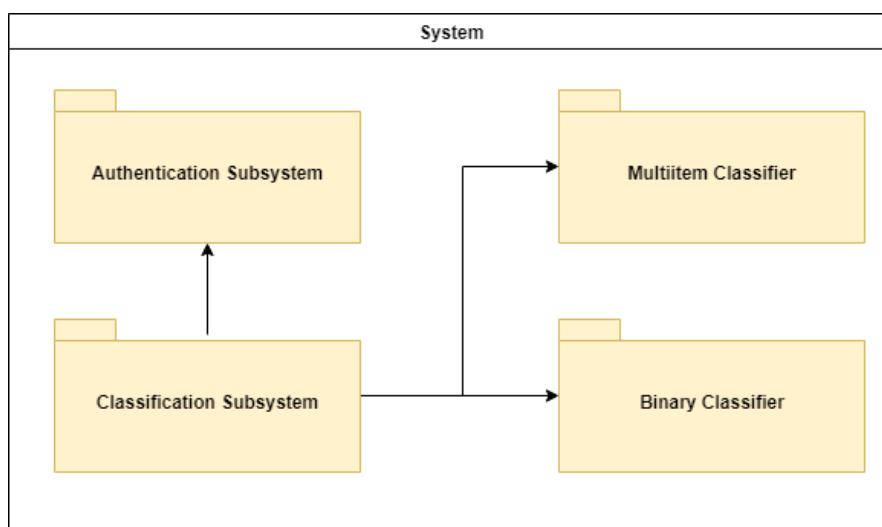


Figure 37 Subsystem diagram

The functionalities of the system are covered in those subsystems. The Classification Subsystem acts as a hub where all the operations are distributed to other subsystems.

5.3.1 Subsystems description

We will describe now the subsystems identified in the previous diagram:

- **Classification Subsystem.** It is the main area of interaction with the user. It communicates with the actual classifying systems in order to achieve the prediction. It includes operations like saving the results to a file and correcting prediction results.
- **Authentication Subsystem.** It is mandatory in order to run administrator operations. It communicates with the classification subsystem. It interacts with the database of registered users.
- **Multiitem and Binary Classifiers.** They constitute two separate subsystems that accomplish the same task differently. They contain training and predicting operations.

5.3.2 Interfaces between systems description

The subsystems communicate internally within the system, without any need of cloud network or external connexion.

- The Authentication Subsystem will forward the administrator user to the Classification Subsystem so the later can show the corresponding features to the administrator. In the case that an authentication is unsuccessful, this subsystem will respond accordingly with the options available for non-administrator users.
- The Classification Subsystem will interact with both the Binary and Multiitem Subsystems for making a prediction of an input text. The correction run is also sent to these subsystems to make their respective training.

5.4 Initial class diagram in analysis phase

Now, the most essential aspects of the class identified in the system are shown, along a brief description of them.

5.4.1 Class diagram

One of the key concepts of the system is the maintainability of it, and so, the addition of a new model should be kept as easy as possible. The strategy design pattern is suitable for this situation. This pattern allows a context class (`ClassificationModule`) to perform the same operation (prediction, correction, training...) in different ways. A Classifier interface establishes the structure and functionalities a model must implement to be used in this application [36].

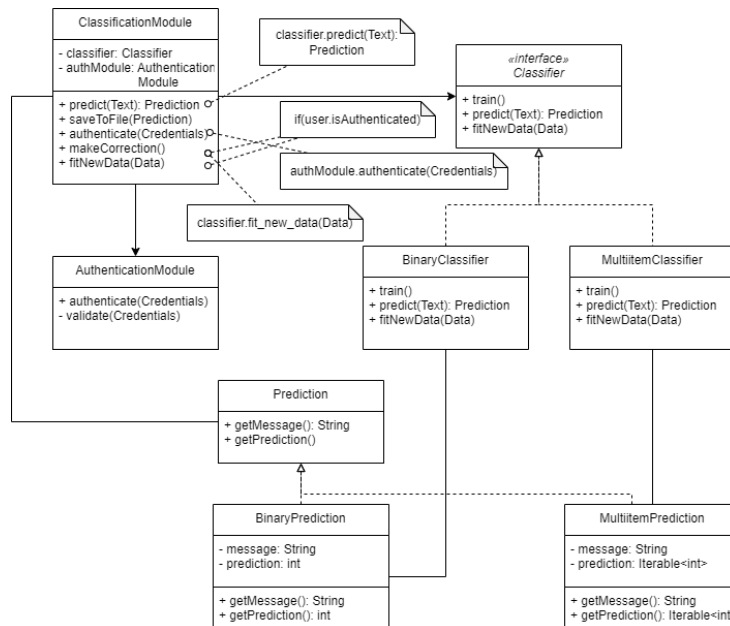


Figure 38 Class diagram in analysis phase

5.4.2 Classes description

Next, we have the descriptions of the classes grouped by subsystem.

5.4.2.1 Classification Subsystem

Class name
ClassificationModule
Description
Gather the user options, both for non-administrator and administrator users
Responsibilities
Make predictions, save results to files, authenticate a user, make a correction of a classification and input new data to train the models. The last two are only available for administrators
Proposed attributes
classifier : Interface that gathers prediction and new data fitting operations authModule : Subsystem to permit authentication
Proposed methods
predict : Takes the text introduced by the user and calls the corresponding method of the classifier attribute. Returns the prediction of the classifier saveToFile : Given a prediction, dumps the message and prediction to a file authenticate : Allows an non-administrator user to log in as an administrator an opt for more advance features makeCorrection : Only for administrators. It asks for the presumed true prediction and fits the data in the model fitNewData : Receives the data to be fit into the corresponding model

Figure 39 ClassificationModule class

5.4.2.2 Authentication subsystem

Class name
AuthenticationModule
Description
Operations of authentication for users
Responsibilities
Authenticate administrators, validate credentials
Proposed methods
authenticate: Asks for the credentials input by the user and grants access if data is valid and correct
validate: Validates the input credentials, checking it is not blank

Figure 40 AuthenticationModule class

5.4.2.3 Binary Classifier

Class name
BinaryClassifier
Description
Operations regarding binary classification
Responsibilities
Training, predicting and fitting new data
Proposed methods
train: This operation is already fulfilled in the classifier. It fits the data and trains the model. Later validation is done to measure the performance.
predict: Takes the text introduced by the user and calls the corresponding method of the classifier attribute. Returns the prediction of the classifier
fitNewData: Receives the data to be fit into the model

Figure 41 BinaryClassifier class

Class name
MultiitemClassifier
Description
Operations regarding itemized classification
Responsibilities
Training, predicting and fitting new data
Proposed methods
train: This operation is already fulfilled in the classifier. It fits the data and trains the model. Later validation is done to measure the performance.
predict: Takes the text introduced by the user and calls the corresponding method of the classifier attribute. Returns the prediction of the classifier
fitNewData: Receives the data to be fit into the model

Figure 42 MultiitemClassifier class

Class name
BinaryPrediction
Description
Wrapper for a message and its binary prediction
Responsibilities

Returning the message and prediction
Proposed methods
getMessage(): Returns the value of the message getPrediction(): Returns the value of the prediction, in this case a single number value

Figure 43 BinaryPrediction class

Class name	
MultiitemPrediction	
Description	
Wrapper for a message and its multiitem prediction	
Responsibilities	
Returning the message and prediction	
Proposed methods	
getMessage(): Returns the value of the message getPrediction(): Returns the value of the prediction, in this case a list with the number values	

Figure 44 MultiitemPrediction class

5.5 Use cases analysis and scenarios

In this section, all use cases and possible scenarios are described, along their preconditions, actors involved, and possible exceptions.

5.5.1 Change classification method

1. Change classification method	
Preconditions	None
Postconditions	Scenario 1.1 The system will prepare any following input text to be predicted against the corresponding model chosen. This step is mandatory to perform a classification
Actors	Non-administrator user or administrator
Description	The user is presented the main window of the application The user selects the option Change classification method The user chooses one of the options
Variation (secondary scenarios)	Scenario 1.2 The user selects an invalid option this problem is dependent of the actual selection mechanism). The system must control the possible error and allow the user only to pick one of the alternatives
Exceptions	An invalid option is chosen

Figure 45 Change classification method use case and scenarios

5.5.2 Detect inappropriate messages

2. Detect inappropriate messages	
Preconditions	Having a selected classification method
Postconditions	The system will compute the results and present it to the user
Actors	Non-authenticated user or administrator
Description	<p>Scenario 2.1 The user introduces a message The option of uploading a file becomes unavailable The user selects the option of perform the classification The system will predict the result with the corresponding classifier The system will present the results to the user</p>
Variation (secondary scenarios)	<p>Scenario 2.2 The user is presented the main window of the application The user chooses the options of uploading the file The option of introducing a message becomes unavailable The file is parsed. Each line is a message to be classified The system performs the prediction of these messages</p> <p>Scenario 2.3 The user introduces a blank message The system must check this behaviour and forbid and warn the user to perform a classification with blank messages</p> <p>Scenario 2.4 The user introduces a too long message The system must forbid and warn the user to introduce more character than the maximum length established</p>
Exceptions	Blank message Too long message

Figure 46 Detect inappropriate messages use case and scenarios

5.5.3 Save results to a file

3. Save results to a file	
Preconditions	The user must have performed a classification and been shown the results
Postconditions	A new file is generated with information about the prediction
Actors	Non-administrator user or administrator
Description	<p>Scenario 3.1 The user is presented the results of the last performed classification The user selects the options save results to a file The system generates a .csv file and dumps the content of the prediction into the file. Each line represents a message, whether the user chose to introduce a message or set of messages</p>
Variation (secondary scenarios)	None
Exceptions	Exceptions corresponding to the file management

Figure 47 Save results to a file use case and scenarios

5.5.4 Correct predictions

4. Correct predictions	
Preconditions	The user must have performed a classification and been shown the results. Run as administrator
Postconditions	The corresponding classifier fits the new prediction as new data
Actors	Administrator
Description	<p>Scenario 4.1 The administrator is presented the results of the last performed classification The administrator selects the option Correct predictions The system will present to the user a mechanism to introduce the presumed true prediction and submit it to the system The system reports to the user the success of the operation</p>
Variation (secondary scenarios)	<p>Scenario 4.2 The user introduces invalid predictions or invalid number of predictions (this is dependent of the input mechanism) The system must validate the data and forbid the user to make an invalid correction</p> <hr/> <p>Scenario 4.3 The user cancels the operation after selecting the option The system hides the mechanism to correct the predictions</p>
Exceptions	Invalid input predictions

Figure 48 Correct predictions use case and scenarios

5.5.5 Log in as administrator

5. Log in as administrator	
Preconditions	None
Postconditions	The system grants a higher level of permission to the user and allows him to perform more operations
Actors	Non-administrator user
Description	<p>Scenario 5.1 The user is presented the main window of the application The user selects the Log in as administrator option The system displays a new window for entering the username and password The user introduces the username and password The user selects the Enter option The system checks the validity of the data The system proceeds to compare the data with the data inside of the database If a match exists, the system returns the user to the main window and grants him access to new features</p>
Variation (secondary scenarios)	<p>Scenario 5.2 The user cancels the operation after selecting the Log in as administrator option The system will return the user to the main window as a non-administrator user</p> <hr/> <p>Scenario 5.3 The user introduces a blank username or password The system validates this data. It warns the user with a proper message The user remains in the window as a non-administrator user</p>

	<p>Scenario 5.4 The user introduces valid data with no match in the database</p> <p>The system will inform the user. The message will not include which of the pieces of data is incorrect, due to security reasons. It will be the same if the incorrect pieces of data are the username, password, or both</p> <p>The user remains in the window as a non-administrator user</p>
Exceptions	Invalid username or password, non-alphanumeric username, too long username or password, database management problems

Figure 49 Log in as administrator use case and scenarios

5.5.6 Train models

6. Train models	
Preconditions	Run as administrator
Postconditions	The chosen classifier fits the new data that will be considered for new predictions
Actors	Administrator
Description	<p>Scenario 6.1 The administrator selects the option Train models</p> <p>The system displays a new window with the classification methods and an option of uploading a file</p> <p>The administrator chooses the model</p> <p>The administrator submits a file containing the data to be fit</p> <p>The system parses the information to be fit to the corresponding model</p> <p>The system reports to the user the success of the operation</p>
Variation (secondary scenarios)	<p>Scenario 6.2 The administrator cancels the operation after selecting the Train models option</p> <p>The system returns the administrator to the main window</p> <hr/> <p>Scenario 6.3 The administrator submits an invalid file</p> <p>The system validates the file. It warns the administrator of the invalidity of the file</p> <p>The administrator remains in the current window</p> <hr/> <p>Scenario 6.4 The administrator submits a valid file with blank data or wrong prediction values</p> <p>The system validates the file data. It warns the administrator of the invalidity of this data</p> <p>The administrator remains in the current window</p> <hr/> <p>Scenario 6.5 The administrator submits a valid file with the wrong number of pieces of data</p> <p>The system validates the file data. It warns the administrator with a message regarding the problem</p> <p>The administrator remains in the current window</p>
Exceptions	Invalid file, invalid data, exceptions corresponding to the file management

Figure 50 Train models use case and scenarios

5.6 Scenarios – Use cases relation

Next, we have a table which attempts to take a visual glance of the relation between scenarios and use cases.

Use Cases Scenarios	1	2	3	4	5	6
1.1	X					
1.2	X					
2.1		X				
2.2		X				
2.3		X				
2.4		X				
3.1			X			
4.1				X		
4.2				X		
4.3				X		
5.1					X	
5.2					X	
5.3					X	
5.4					X	
6.1						X
6.2						X
6.3						X
6.4						X
6.5						X

Figure 51 Relation between scenarios and use cases

5.7 User interface analysis

The main concern about the user interface is that it must be reduced to its simplest terms. The aim of this project is not to elaborate the most complex interface to explore human-computer interaction. Nevertheless, the interface must follow the adaptability conventions and fulfil basic use experience features.

5.7.1 Interface description

The program will consist of three windows with different goals.

5.7.1.1 Main window

This will be the first and main window of the interface. It encapsulates the basic operations of the program and leads to the other two windows. The options of this window are:

- Selecting a classification method.
- Submitting a message via keyboard.
- Submitting a message via file.
- Logging as administrator option.
- Showing the results of a prediction.
- Clearing the contents to perform another classification.
- Saving the results to a file.
- Correcting a prediction (if user is administrator).
- Train models option (if user is administrator).

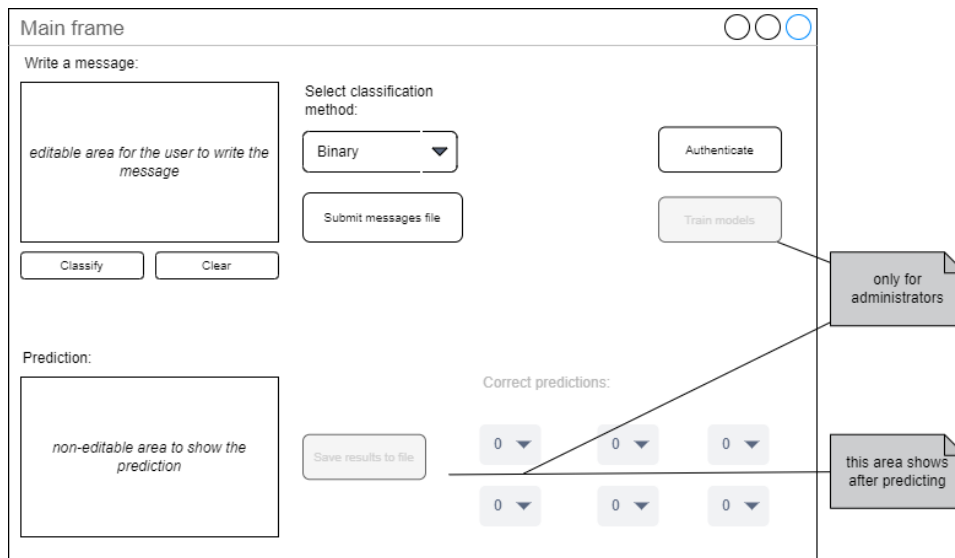


Figure 52 Main window

The following figures represent the main window on several states.

5.7.1.1.1 Start of application

Figure 53 represents the first state of the application.

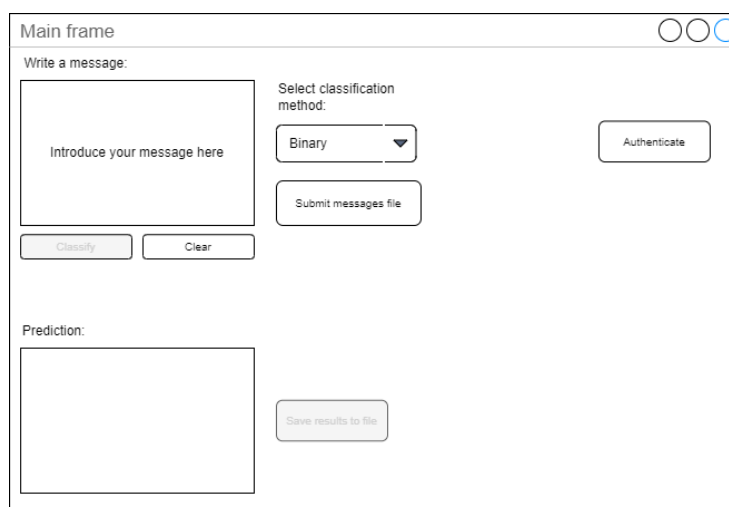


Figure 53 Main window, start of application

5.7.1.1.2 Message typed in area, classification not yet performed, non-administrator user

As the message is introduced, the buttons Submitting messages file and Save results to file become disabled, and the Classify option is enabled.

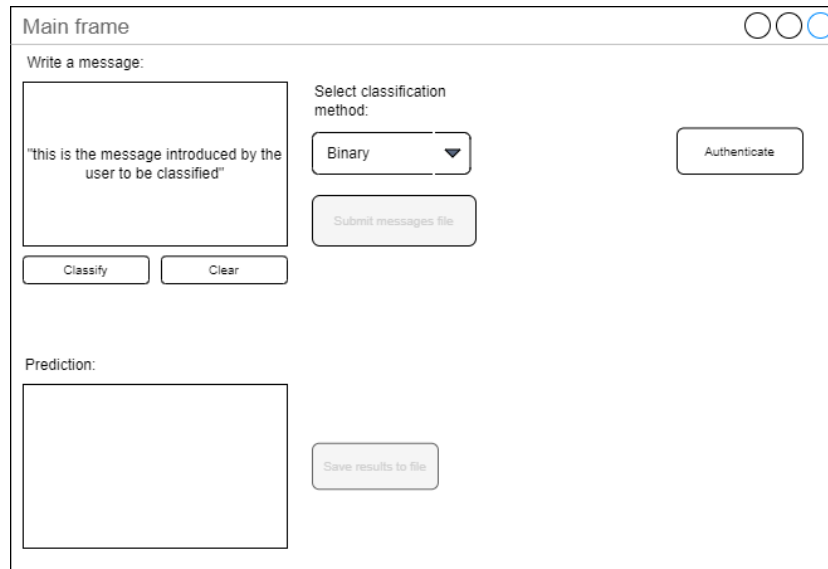


Figure 54 Main window, message typed in area, classification not yet performed, non-administrator user

5.7.1.1.3 Messages uploaded by file, classification not yet performed, non-administrator user

If the message file is submitted instead the text area becomes unavailable. A message confirms the submission.

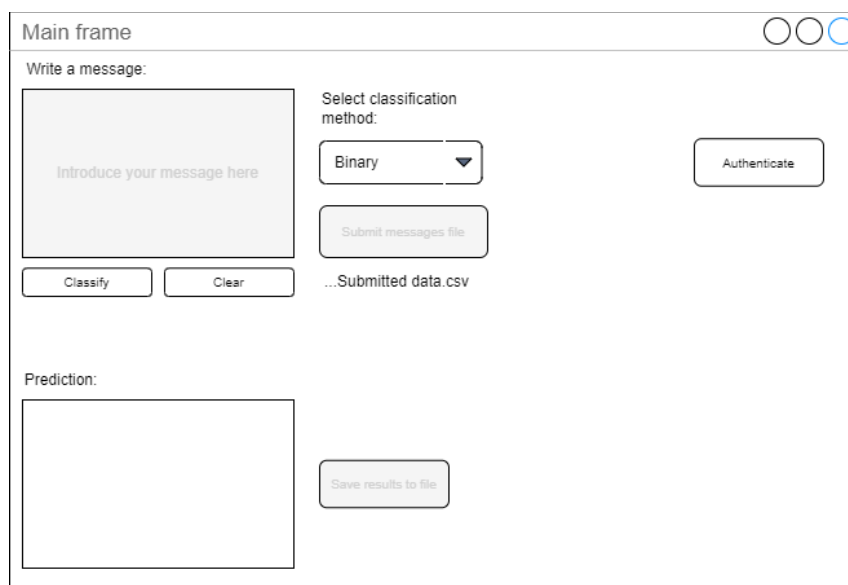


Figure 55 Main window, message uploaded by file, classification not yet performed, non-administrator user

5.7.1.1.4 Classification performed, non-administrator user

When the classification is performed, both message input options become disabled, and the second text area shows the results. The Save results to file option is available now.



Figure 56 Main window, classification performed, non-administrator user

5.7.1.1.5 Start as administrator

After fulfilling the authentication form in the Authentication window, the main window will look like this. The difference is in the Train models button that is now visible.

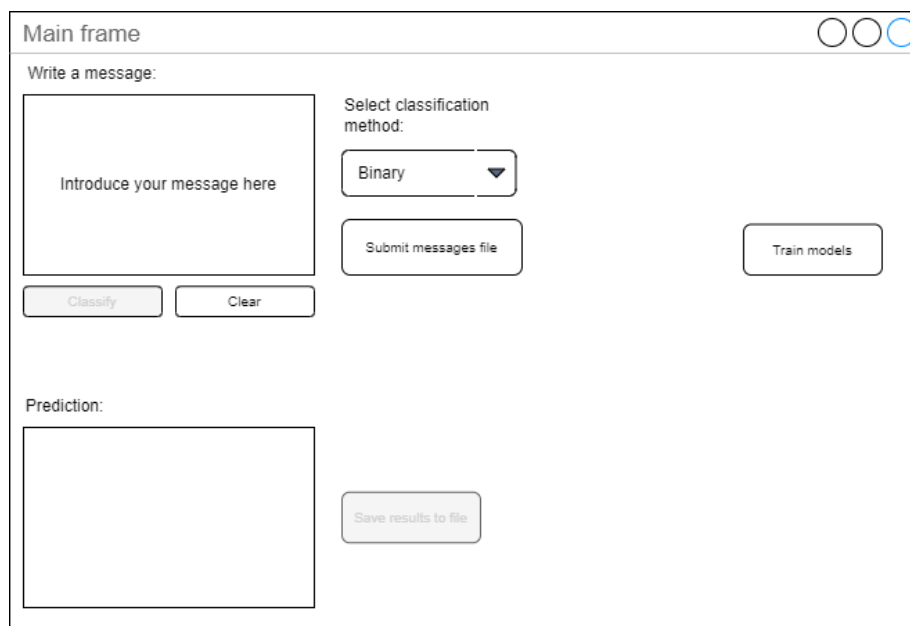


Figure 57 Main window, start as administrator

5.7.1.1.6 Message typed in area, classification not yet performed, administrator

Same as Figure 54, except the Train models option.

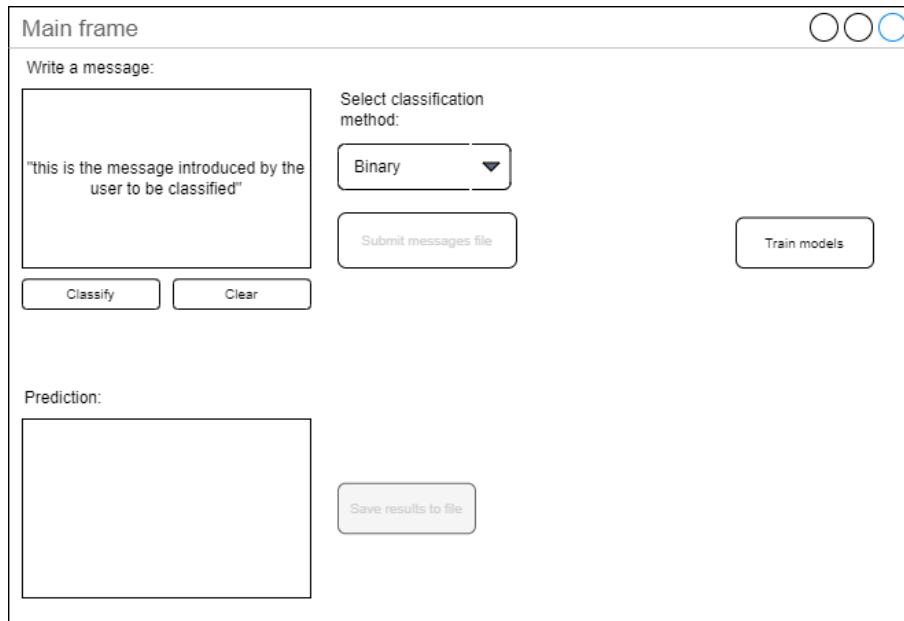


Figure 58 Main window, message typed in area, classification not yet performed, administrator

5.7.1.1.7 Messages uploaded by file classification not yet performed, administrator

Same as Figure 55, except the Train models option.

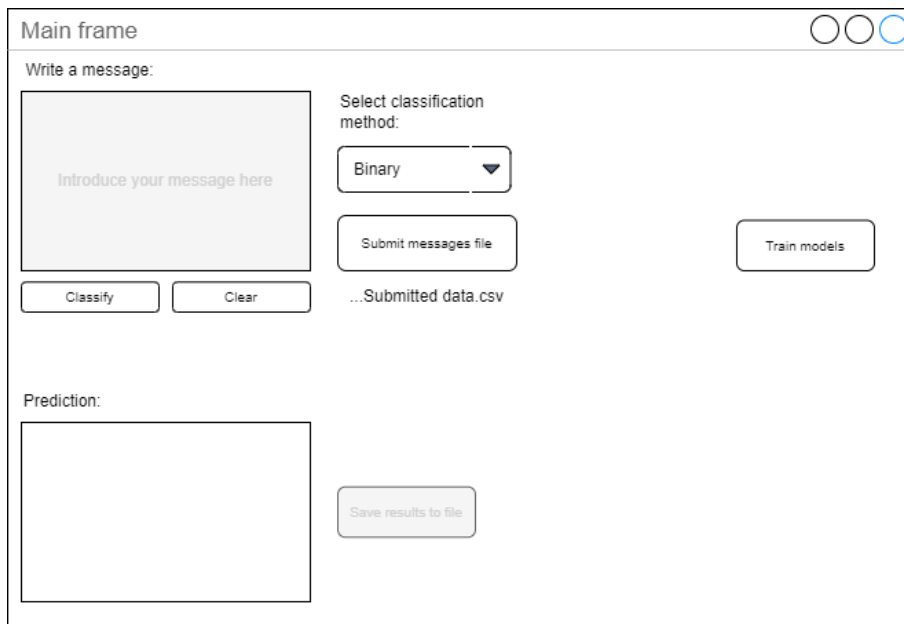


Figure 59 Main window, message uploaded by file, classification not yet performed, administrator

5.7.1.1.8 Classification performed, administrator

When the classification is performed, a new panel in the right bottom corner is shown, with options to select the new predictions for the message.

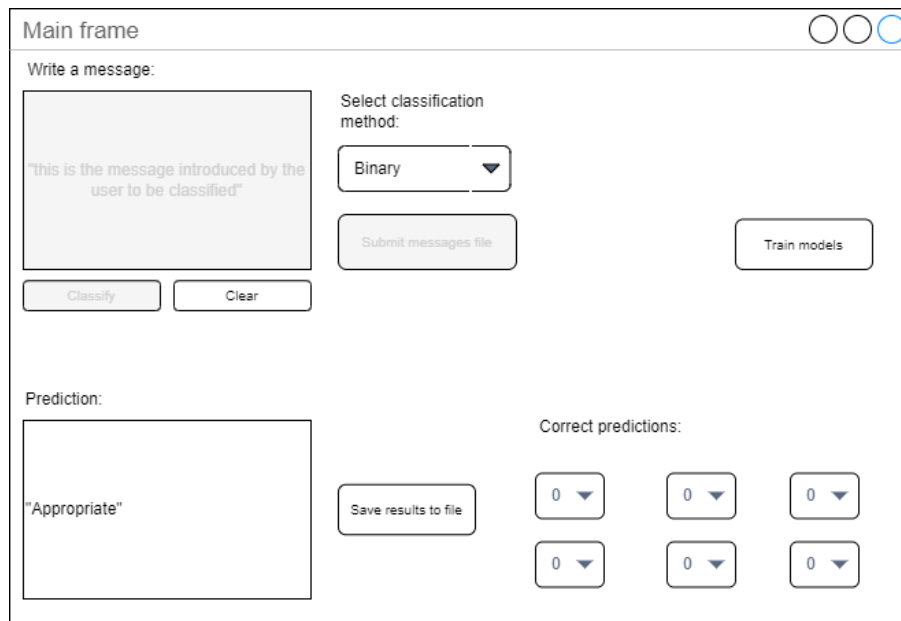


Figure 60 Main window, classification performed, administrator

5.7.1.2 Authentication window

This is the window that manages the inputs of the user when attempting the authentication as administrator.

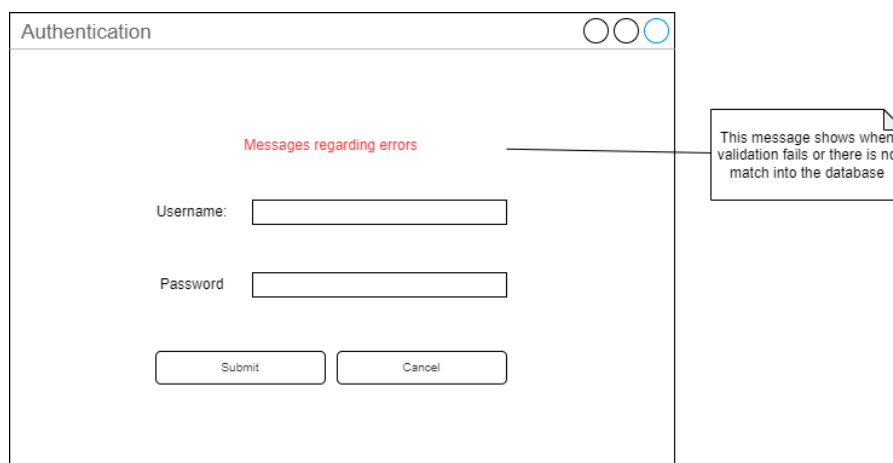


Figure 61 Authentication window

5.7.1.3 Train model window

This window manages the input file that the administrator submits to fit new data to the corresponding classifier.

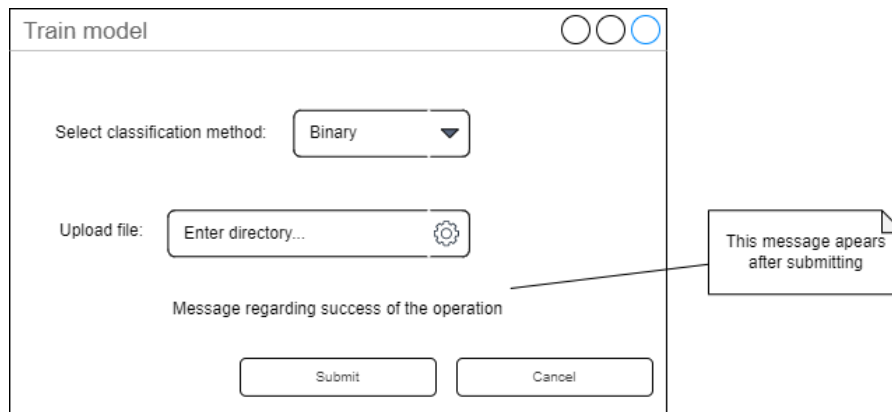


Figure 62 Train model window

5.7.2 Interface behaviour description

This system is heavily influenced by the user inputs, so validation and verification are needed to avoid malfunctioning. Now we will see the messages that the system reports in response to the user possible inputs.

5.7.2.1 Main window

- Blank messages files containing blank messages are not valid inputs. The window will show a message informing the user saying: “Message is not a valid message”. There is no need to validate the contents of the message.
- Files uploaded for predicting its content should only have one column per line, representing the message. The window will display the message “The file does not have the correct number of columns” if it is the case.

5.7.2.2 Authentication window

- Username or password being blank are not valid and would lead to a message of the system report the issue. The message is “Username/password cannot be blank”.
- Username and password cannot exceed their respective maximum length. If a longer input is introduced the system will display “Username/Password has exceeded the maximum number of characters”.
- For valid username and password that do not match any content inside the database the system will display “Incorrect username or password”. It is important not to specify the concrete piece of data that is incorrect, due to security reasons. A person with harmful intentions and with basic knowledge on software could try an attack to the database. If the attacker knows the concrete field that is incorrect, it could focus the attention into this field.

5.7.2.3 Train models window

- The input of the user will be a file containing the messages and prediction values. If the extension is not .csv the message “Input file has wrong extension”.
- There are several validation steps for a valid file. For blank messages, wrong number of categories and wrong prediction values will be reported to the user with the message “The prediction of message is not valid for fitting” and the actual data error.

5.7.3 Navigability diagram

Analysed all the windows of the system, the navigability of the windows would look like Figure 63.

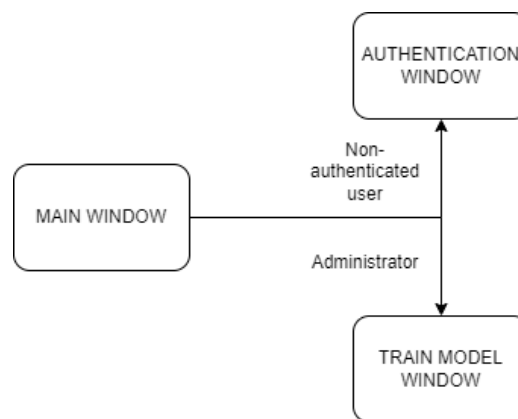


Figure 63 Navigability diagram

5.8 Test plan specification

In this subchapter we will discuss the test plan itemized in five categories: unitary, integrations, system, usability, and code testing.

5.8.1 Unitary testing

Unit testing will be used to validate and verify the correct functionality of the modules of the system. It is key to test the scenarios previously detailed, both the expected correct user interaction and the error-prone ones.

5.8.2 Integration testing

For integration testing, the system will be subdued to correct relations between subsystem assertion. The authentication submodule should correctly grant access to a user, so the application shows the correct options to the user.

5.8.3 Usability testing

The system will be put in review to third-party users which will use the application both as a non-administrator user and administrator. Their opinions on the system will be compiled with some surveys.

5.8.4 Performance testing

There will be tests on prediction and training computation speed and memory usage.

5.8.5 Use cases testing

<i>Use case 1: Change classification method</i>	
Test 1.1	Expected Result
The user changes the method to any of the two possible methods	The system will prepare the corresponding classifier
Test 1.2	Expected Result
The user changes the method to a not contemplated method	The system does not register the change and informs the user of the error

Figure 64 Change classification method use case test

<i>Use case 2: Detect inappropriate messages</i>	
Test 2.1	Expected Result
The user tries to classify a blank message with no file uploaded	The system informs the user of the error/The user interface disables the Classify option
Test 2.2	Expected Result
The user tries to classify a message that exceeds maximum message length	The system informs the user of the error/The user interface prevents the excess of characters typed by the user
Test 2.3	Expected Result
The user tries to upload a file with wrong extension	The system catches the error and warns the user about it
Test 2.4	Expected Result
The user tries to upload with unexpected data	The system catches the error and warns the user about it
Test 2.5	Expected Result
The user types a message and tries to upload a file with messages	The system discards the file submission and warns the user/The user interface disables the Upload file option
Test 2.6	Expected Result
The user uploads a file and tries to type a message	The system discards the input message and warns the user/The user interface disables the area to type messages
Test 2.7	Expected Result

The user types a message with no file uploaded and classifies the message	The system sends the message to the classifier, which will return the prediction that will be shown to the user The Save results to file options becomes available If the user is an administrator, the option Correct predictions becomes available
Test 2.8	Expected Result
The user uploads a correct file with valid data and with no typed message	The system sends the messages (one per file line) to the classifier, which will return the predictions that will be shown to the user The Save results to file options becomes available If the user is an administrator, the option Correct predictions becomes available

Figure 65 Detect inappropriate messages use case test

Use case 3: Save results to file	
Test 3.1	Expected Result
The user tries to save results to file with no classification performed	The system informs the user of the error/The user interface disables the Save results to file option
Test 3.2	Expected Result
The user saves results of a performed classification to a file	The system automatically creates a file and dumps the information of the message or messages and its/their classification on the file

Figure 66 Save results to a file use case test

Use case 4: Correct predictions	
Test 4.1	Expected Result
The administrator tries to correct a prediction with no classification performed	The system informs the user of the error/The user interface disables the Correct predictions option
Test 4.2	Expected Result
The administrator introduces a correction with wrong values or number of values	The system informs the user of the error/The user interfaces only offers the possible values and the mandatory number of values
Test 4.3	Expected Result
The administrator introduces a correct prediction	The system fits the new data to the model and shows the success of the operation to the administrator
Test 4.4	Expected Result
The administrator cancels the operation of correcting the predictions	The system returns to the previous state and no modification is done
Test 4.5	Expected Result
A non-administrator user tries to correct the prediction	The system checks beforehand if the user is an administrator/The user interface hides this option

Figure 67 Correct predictions use case test

Use case 5: Log in as administrator	
Test 5.1	Expected Result

The administrator tries to access the authenticate option	The system reports the user that it is already authenticated as administrator/The user interfaces disables the Authenticate option
Test 5.2	Expected Result
The user selects the option Log in as administrator	The system asks to the user for the username and password
Test 5.3	Expected Result
The user tries to introduce blank username or password	The system communicates the error to the user/The system disables the Submit option
Test 5.4	Expected Result
The user tries to introduce too long username or password	The system communicates the error to the user/The system disables the Submit option
Test 5.5	Expected Result
The user tries to introduce non-alphanumeric characters for the username	The system communicates the error to the user/The system prevents the user from typing non-alphanumeric characters
Test 5.6	Expected Result
The user introduces valid username and password but there is not match in the database	The system communicates to the user that the username or password does not exist
Test 5.7	Expected Result
The user introduces valid and correct username and password	The system grants administrator access to the user and returns it to the main window
Test 5.8	Expected Result
The user cancels the Log in as administrator operation	The system returns the user to the main window with no administrator access

Figure 68 Log in as administrator use case test

<u>Use case 6: Train models</u>	
Test 6.1	Expected Result
The non-administrator user tries to train a model	The system informs the user about he has no access to that operation/The user interface disables the Train models option
Test 6.2	Expected Result
The administrator selects the Train model option	The system asks the model to be trained and the file containing the data
Test 6.3	Expected Result
The administrator tries to upload an invalid file	The system shows the error to the user/The system disables the Train option
Test 6.4	Expected Result
The administrator uploads a file with blank data or wrong prediction values	The system shows the error to the user
Test 6.5	Expected Result

The administrator uploads a file with wrong number of pieces of data	The system shows the error to the user
Test 6.6	Expected Result
The administrator uploads a valid file with valid data and trains the model	The system will parse the data and pass it to the classifiers that will fit it and consider it to following predictions
Test 6.7	Expected Result
The administrator cancels the operation	The system returns the user to the main window

Figure 69 Train models use case test

Chapter 6. System design

This chapter discusses topics from the design phase of this project, the system architecture, shows several diagrams representing the system structure and behaviour, database and user interface design and the technical specification of the test planning.

6.1 System architecture

This section covers the package distribution and component integration of the system.

6.1.1 Package diagram

The following diagram shows the packages identified in the system and their relations. The code of colours is the following:

- Green: For the user interface and output files from saving the results to files. We may see they have heavy interaction with the user.
- Pink: Domain package.
- Gray: Packages with raw data.
- Blue: Auxiliar packages. They contain operations shared by several packages.
- Red: Business logic and calculations are inside these packages.

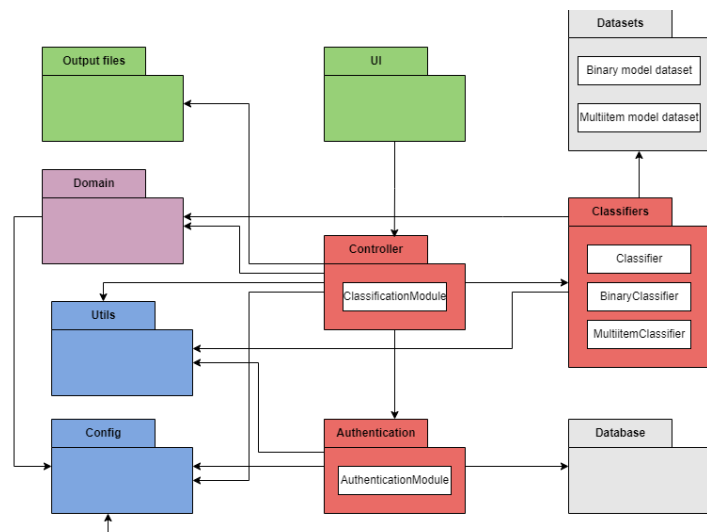


Figure 70 Package diagram

6.1.1.1 Controller

This package will contain the logic of all the operations a user may do, acting like a controller which will distribute responsibilities to other modules. The ClassificationModule is considered a

class as seen in the chapter 5.4, but it may be split up as more classes which would be more cohesive.

6.1.1.2 Authentication

The authentication package will handle all the operations regarding the logging of a non-administrator user. This package is the only one which will access to the database containing users' data.

6.1.1.3 Classifiers

This package contains the operations related to the models, like training, predicting, and fitting new data. Note that although different systems, the two classifiers are encapsulated in the same package.

6.1.1.4 Utils

An auxiliar operation package. It takes responsibility of tasks related to several packages like managing files and low-level operations like pre-processing text data. It must not contain any business logic regarding system requirements.

6.1.1.5 Config

Package containing configuration files which contain constants, output formatted messages, etc, in addition to the logging configuration. This package is used by several other packages.

6.1.1.6 Database

It will contain the actual database which hold the users' information.

6.1.1.7 Datasets

This package will contain the data used to train and test the models.

6.1.1.8 Domain

This package contains domain-related classes. For example, a Prediction class would be included inside this package.

6.1.1.9 Output files

Used for grouping all the files that the user may generate from the option Save results to file when performing a classification.

6.1.1.10 User interface

It contains the classes of the user interface. This class should be as low-coupled as possible, given that the user interface can be subject to a large quantity of changes or be replaced by, for instance, a command-line interface.

6.1.2 Component diagram

The component diagram of this project is displayed now.

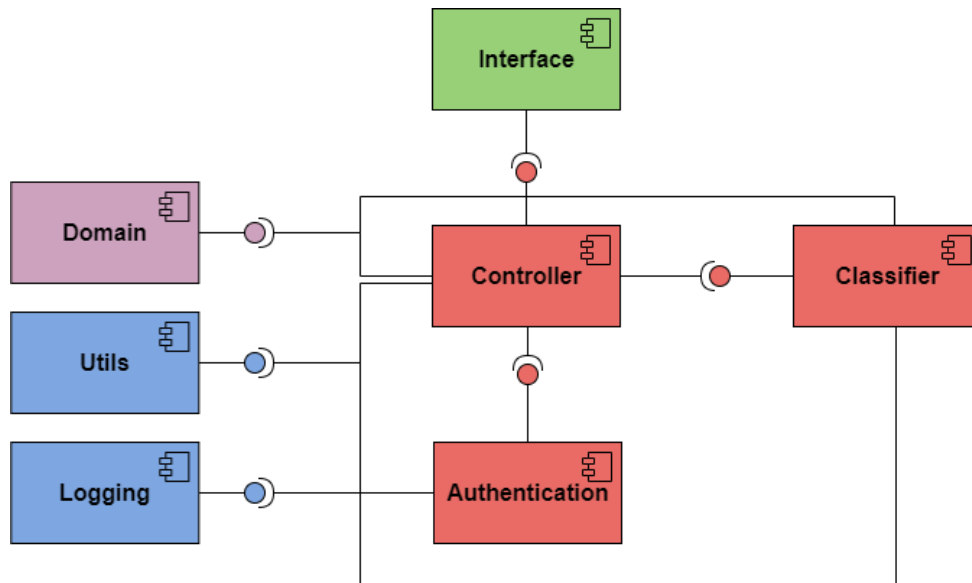


Figure 71 Component diagram

The code of colour is the same as the section above. We can see the connections between components and which interfaces are needed for each component. Auxiliar components will offer operations shared by the logic classes. The user interface will also take use of the Controller component interface. The domain component will be used in the controller and classifier ones.

6.2 Class design

Now we will see the classes designed categorized in their respective packages.

6.2.1 Controller package

For this package only one class has been designed for this package.

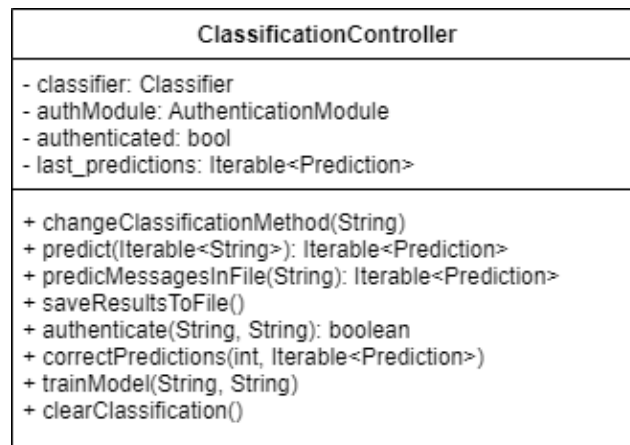


Figure 72 Controller package class diagram

The ClassificationController class acts as a hub where are the operations of the user are compiled and processed. This class assigns the operation to the corresponding modules. Note that the Strategy pattern is applied regarding the classification method. This will be commented in the Global class diagram subsection [36]. The attributes for this class are:

- classifier: An object of the class that contains the operations of the model, like training, predicting, and fitting new data. It will change whenever the user uses the option Change classification method.
- authModule: This object corresponds to the class that manages all the authentication process.
- authenticated: Describes whether the user is an administrator or not.
- last_predictions: The last predictions performed in the system.

The methods of the class are:

- changeClassificationMethod(String): Changes the classifier based on the model option passed as a String parameter.
- predict(Iterable<String>) → Iterable<Prediction>: Predicts a set of messages and returns their predictions.
- predictMessagesInFile(String) → Iterable<Prediction>: Predicts a set of messages. The String represents the path of the file introduced by the user.
- saveResultsToFile(): Creates a file in an output folder and saves the content of the last_predictions attribute.
- authenticate(String, String): Calls AuthenticationModule to perform the operation. The returned Boolean represents whether the user has been authenticated or not.
- correctPredictions(int, Iterable<Prediction>). Receives the index of the prediction and the list of new predictions to train model. Only available for administrators.
- trainModel(String, String). Trains the model represented by the String parameter. The String corresponds to the path of the new data to be fit.
- clearClassification(). Prepare the system to perform another classification.

6.2.2 Authentication

Two classes pertain to this package.

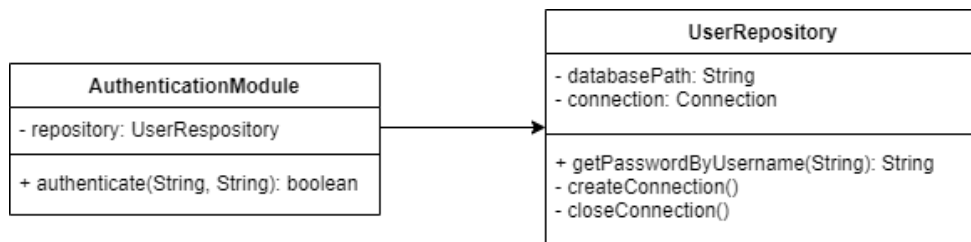


Figure 73 Authentication package class diagram

The `UserRepository` class is responsible of doing the low-level database operations, like creating and closing the connection and creating and executing the query. The attributes are:

- `databasePath`. Path for the connection to find the database.
- `connection`. A connection that will be created and closed in every query.

The methods are:

- `getPasswordByUsername(String)`. The main operation of the class. It creates a query that demands the password of the username given.
- `createConnection()/_closeConnection()`. Private methods.

The `AuthenticationModule` class contains one `UserRepository` attribute to perform the operations described above. The method of this module receives the password from the username provided and checks, with the proper hashing algorithm, if the introduced password matches the hashed password of the database.

6.2.3 Classifiers

The Classifiers package consists of one interface and two classes implementing it.

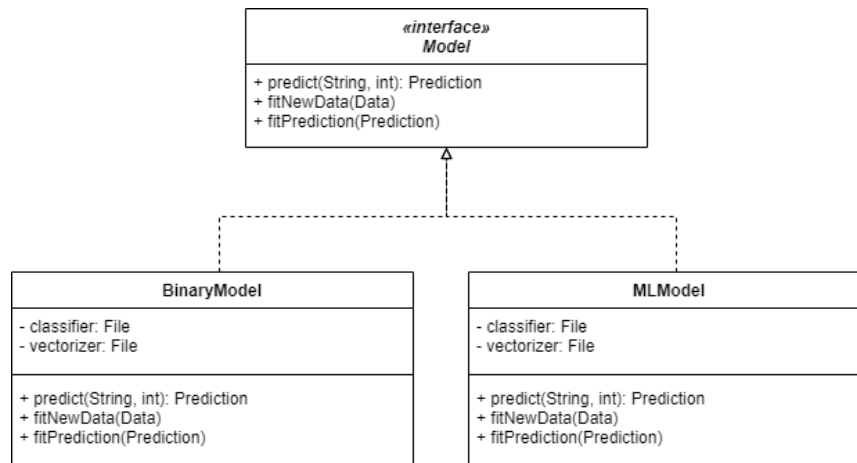


Figure 74 Classifiers class diagram

The Model interface specifies a contract upon which the Controller package can agree. The methods of this interface are:

- `predict(String, int)`: The invoked method by the Controller package.
- `fitNewData(Data)`: It fits the model with new data that will be considered for following classifications. Both Iterable are for the messages and predictions.
- `fitPrediction(Prediction)`: Receives a Prediction to be fit.

The classes implementing the interface do the same operations but may do it in different ways. The attributes in both classes are intrinsic attributes to these models to perform the classifications.

6.2.4 Utils

This package will contain several files used in a major part of the system to perform common tasks:

- `file_management.py`: Used for file dumping, opening, closing and conversion to specific data structures.
- `nlp.py`: For natural language processing tasks like removing links, Twitter usernames and hashtags, lower-casing, etc.
- `validation.py`: Used for validation user inputs.
- `stats.py`: Computes statistics for a classifier given validation data.

6.2.5 Config

Contains the configuration for the system and the logging

- `config.py`: Declares constants, directories, queries, and output messages. It useful if changes are required.

- `logconfig.py`: The logging configuration used for the different levels of logging are explicitly stated here.
- `uiconfig.py`: Contains messages for the user interface and the identifiers of the window elements.

6.2.6 Domain

An interface with two implementing classes were considered for the domain package.

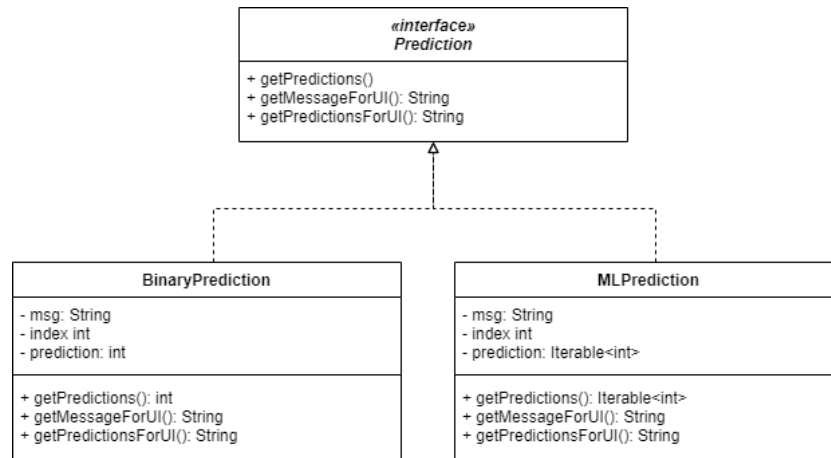


Figure 75 Domain class diagram

The prediction interface encapsulated all the information needed to be passed from the Classifier component to the Interface one. The methods are:

- `getPredictions()`: These methods return the raw values of the prediction of the corresponding classifier.
- `getMessageForUI() → String`: Method used for obtaining a formatted message suitable to the user interface.
- `getPredictionsForUI() → String`: Method used for obtaining a formatted output suitable to the user interface.

The classes implementing the interface will alter the `getPredictions()`, which will return different types. The `BinaryPrediction` class will return a single `int` value for the prediction whilst the `MLPrediction` class returns a `Iterable<int>`, each position corresponding to an item. Likewise, it happens the same for the `prediction` attribute.

6.2.7 User interface

The user interface will consist of three main classes, representing each window.

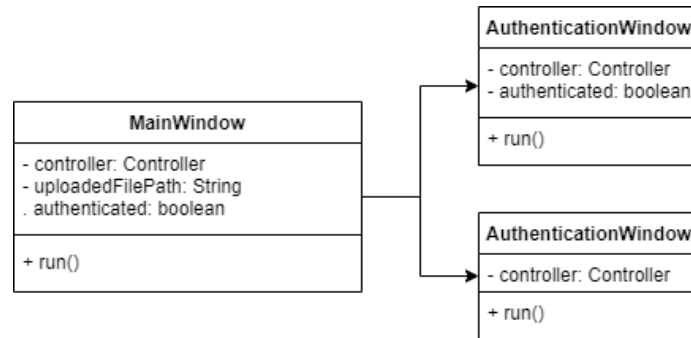


Figure 76 User Interface class diagram

The MainWindow class will contain the Controller object that receives the user interaction petitions, a String representing the submitted file and authentication check. The run () method will make the display of the frame. In addition, it will include a AuthenticationWindow and a TrainingWindow, used when the corresponding option is selected. These classes also contain their particular run () method. AuthenticationWindow has an authenticated Boolean attribute to determine the authentication.

6.2.8 Global class diagram

Thus, the following global class diagram is resolved.

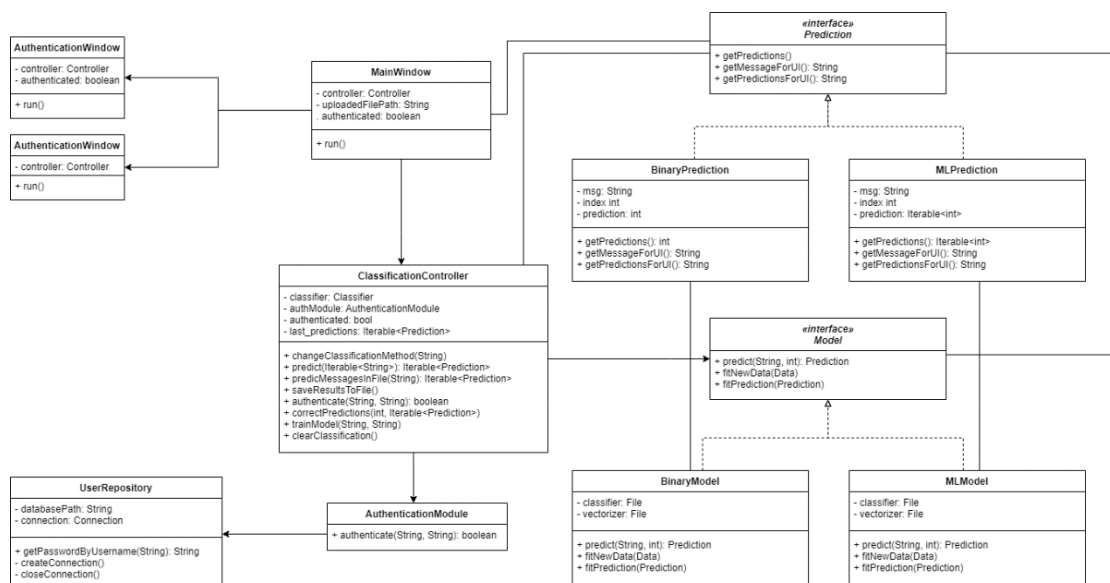


Figure 77 Full system class diagram

The main functionality of the system is designed with a Strategy pattern [36]. The `ClassificationController` acts as the context and the Model is the actual strategy, that is implemented by two concrete strategies. The system attempts to achieve high cohesion and low coupling in every class. If there is a decision of changing a particular model, or just simply adding another one, it would only have to implement the `Model` interface and supply the agreed methods, and maybe implement an additional `Prediction` class, depending on the case.

6.3 Interaction diagrams

This section will show visually the flow of interaction in the use cases available, and a brief explanation of each operation.

6.3.1 Detect inappropriate messages

The use case of detecting inappropriate messages is shown below in an interaction diagram.

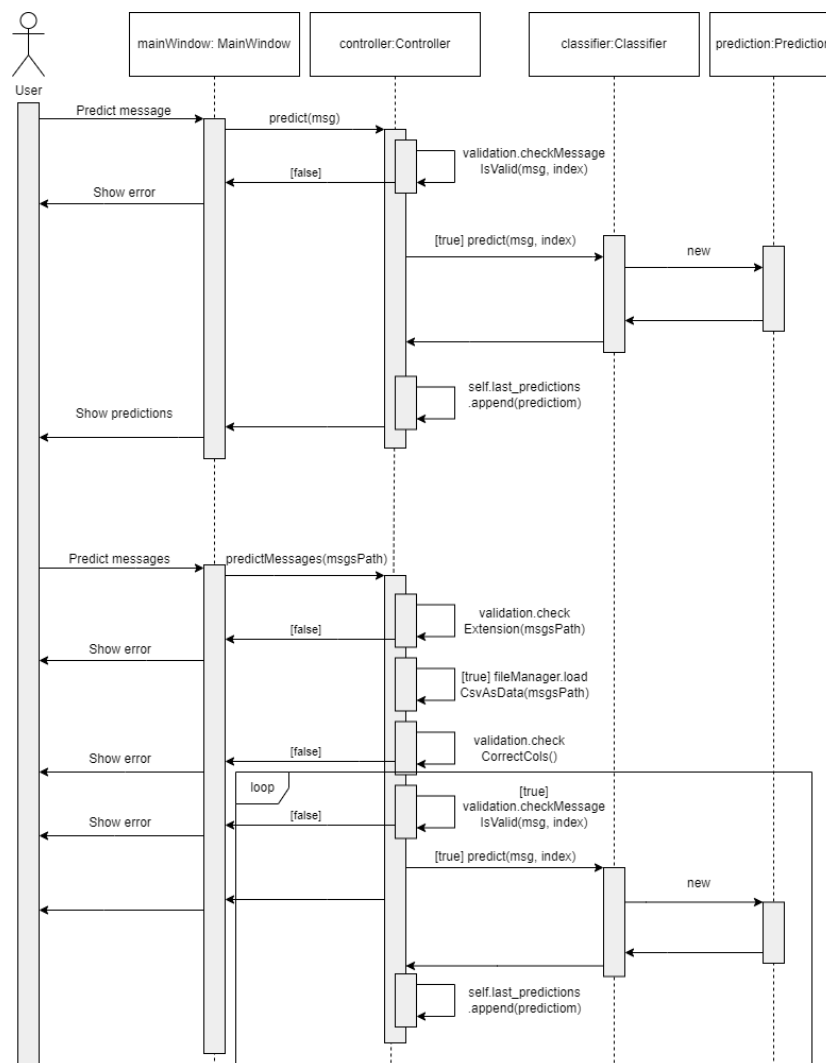


Figure 78 Detect inappropriate messages use case interaction diagram

Both options of detecting inappropriate messages are described, and their possible ramifications.

The user types a message and selects the Predict option. This event is captured in the window, which will transfer the task to the controller. Then, the controller validates the input message and, if it is not valid, will return to the window to display the issue to the user. If it is a valid message, the message is forwarded to the corresponding classifier, which will be in charge of creating the prediction object. This prediction is then returned to the window to be displayed not before the controller adds it to the `last_predictions` attribute.

The input of several messages through a file also passes through the controller. It will validate the file format and will convert it to a manageable data structure. Otherwise, it will return to the window to inform the error. Then it validates the number of columns and if this operation fails the error will be also displayed. If it does not, a prediction for each message will be computed, as the ones described before.

6.3.2 Train model use case

The use case of training a model is shown below in an interaction diagram.

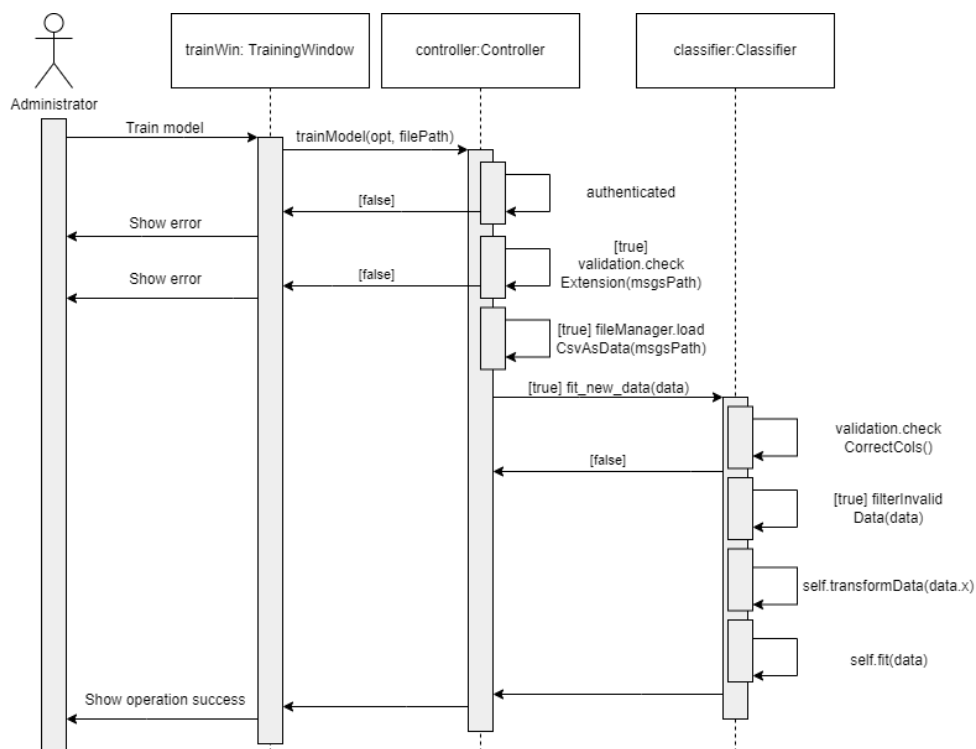


Figure 79 Train model use case interaction diagram

This is the process flow of training a classifier. The administrator selects the option Train model. The event is passed to the controller which will check first the authentication. If the user is authenticated, an error is passed to the user interface. If not, the validation follows to check the file extension, which, in the case it is incorrect, will be reported to the user. Next, the file is converted to a data structure and passed to the classifier to be fit. The classifier examines the

validity of the data. This operation is done inside the classifier and not in the controller because the classifier has the necessary information for approving the data. First, the number of columns is checked. If the number is incorrect, the present error is forwarded to the user. If it is correct, the classifier will verify and filter the message and prediction of each piece of data and inform the user of the possible error. Then, the classifier transforms the data to its proper format and performs the training. After this, the training is completed.

6.3.3 Correct predictions use case

The use case of correcting predictions is shown below in an interaction diagram.

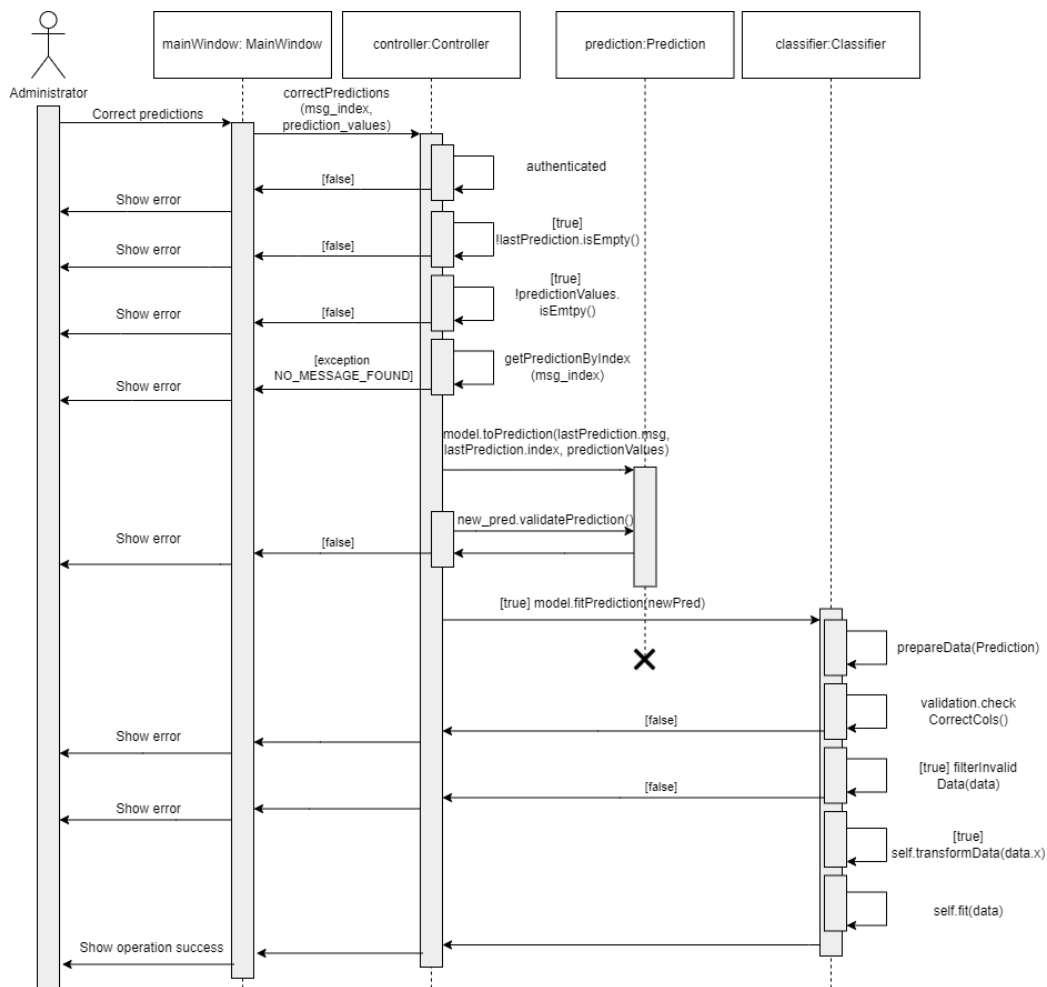


Figure 80 Correct prediction use case interaction diagram

This use case uses a similar approach to the Train model use case process flow. When the administrator selects this option in the interface, the captured event is processed by the controller, which will be verify if the user is, indeed, and administrator. The program will return to the user interface to inform the user if it is not. If it is, the next step is to check if a prediction has been performed yet. It has not, the error will be prompted to the user and if it is, the program examines if the input prediction is empty. If it is empty, an error is reported to the administrator. Now the controller checks for the message given the index provided as parameter. If it does not

exist in the `last_predictions` attribute, it will raise an exception to inform the user. The system will transform the prediction to a `Prediction` object to be passed to the model. The model then transforms the prediction to a suitable data structure to manage, and this structure is passed to the process of the previous use case.

6.3.4 Log in as administrator use case

The use case of authenticating is shown below in an interaction diagram.

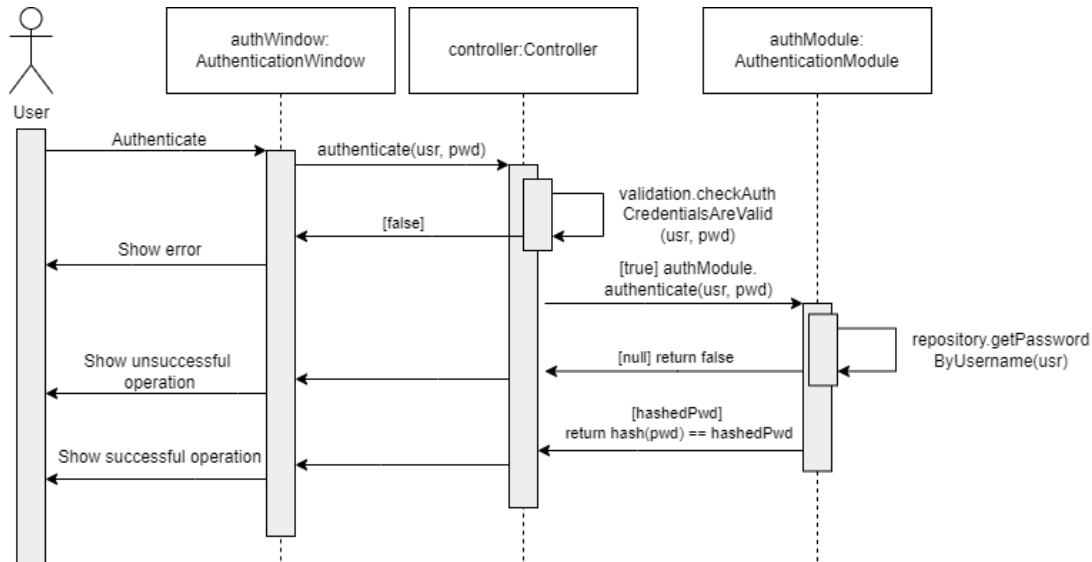


Figure 81 Log in as administrator use case interaction diagram

The authentication process is a simple one. The user introduces the username and password and selects the submit option. The window catches the event who will be forwarded to the controller. The controller checks the validity of the user inputs, both username and password and it will report an error to the user if this validation fails. This validation consists of assertions of length blank credentials, too long credentials, and a non-alphanumeric username. Then, the information is passed to the authentication module, whose repository will query the password of the given username. If nothing is returned, the operation was unsuccessful, and the result is shown to the user. If there exists a password for that username, the module will return the true value for the comparison between the hash of the input password and the one retrieved.

6.3.5 Change classification method use case

Since this use case is much simpler than the described above, there is no need of an interaction diagram. The steps are explained as follows:

- The user selects the option Change classification method and changes the value of it. This event is captured by the window.
- It is then passed to the controller.
- If it is a wrong value, the system will warn the user and not alter the state.

- If it is a good method value, the system will change the classification method to the required one.

6.3.6 Save results to file use case

Likewise, this use case is not as complex as the first three use cases, so an explanation of the process flow is explained now:

- The user selects the option Save results to file. The event is captured by the window.
- The event is passed to the controller.
- If a prediction did not exist in the first place, the system reports the issue to the user.
- If a prediction did exist, the system will use the file manager to create the output file and dump the content of the prediction.

6.4 System persistence

The system makes use of two machine-learning models for the binary and the itemized classification respectively. These models are stored inside files with extension *.pkl*, which stands for Pickle, a common Python serialization module that allow to store serialized Python objects [37].

When a prediction is corrected or when an administrator trains a model, the operation ends with the dumping of the new model to these Pickle files.

For the authentication module, the administrators are already stored in a local database.

6.4.1 Database design

Now, the decision of the DBMS is reasoned. Also, the Entity-Relationship diagram is shown.

6.4.1.1 Used DBMS description

The Database Management System chosen is SQLite. It is optimal for our system, given the simplicity of the database content. SQLite implements a small, fast, and self-contained database engine [38].

6.4.1.2 DBMS system integration

The local database is managed through a class inside the Authentication package called `UserRepository`. This class is responsible of creating the connection, making the necessary queries, and closing the connection. Every query is self-contained, which means that it creates and closes the connection within itself.

6.4.1.3 E-R Diagram

Given that only users are stored inside the database, there exists only one entity.

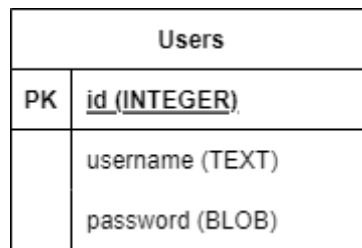


Figure 82 Database Entity-Relationship diagram

The id acts as the primary key, and username and password are the information stored. The password is stored as BLOB, since a hash is stored for every tuple.

6.5 Interface design

These are the final prototypes of the user interface for every window. For the most part, they are identical to the analysis design. However, a new confirmation window for administrator operations has been added.

6.5.1 Main window

This would be the main window interface.

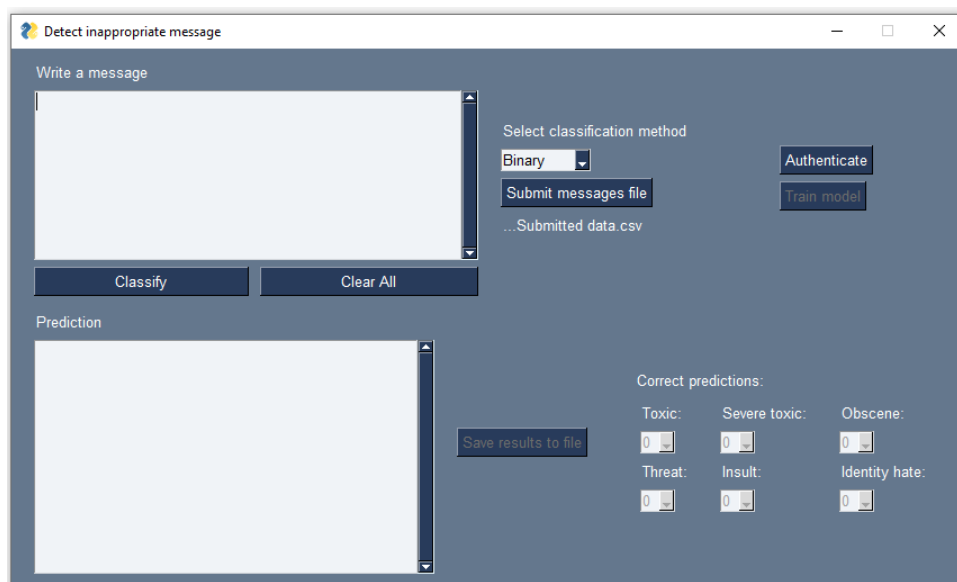


Figure 83 Final Main Window

The main window will look like this. There might be a few changes about buttons positioning, and text font and colours.

6.5.2 Authentication window

The authentication window looks like this.

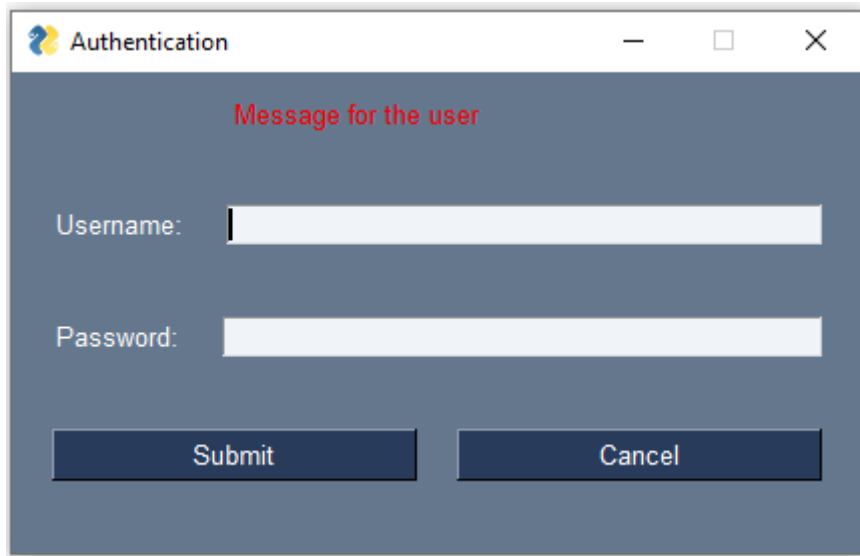


Figure 84 Final Authentication Window

This is the final design for the Authentication window. Some features may change, for instance the font, colour, and position of the text.

6.5.3 Training window

Figure 85 shows the Train model window.

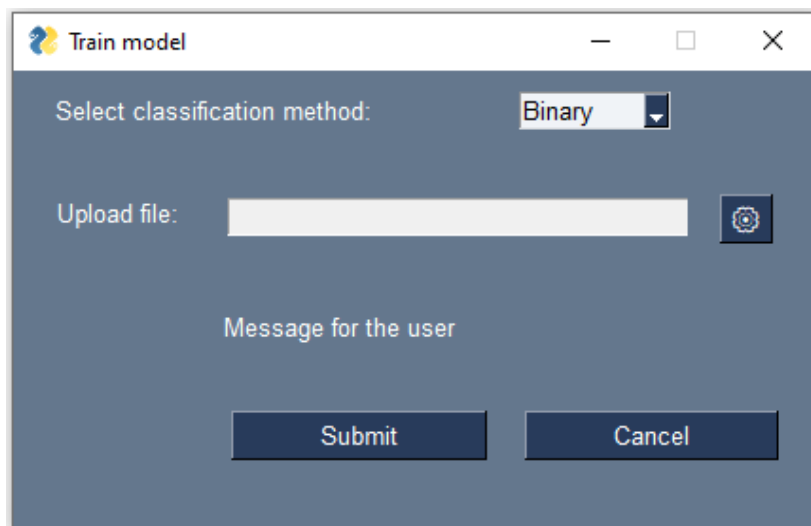


Figure 85 Final Training Window

6.5.4 Confirmation window

For the administrator operations of correcting a prediction and training a model, a new confirmation dialog has been added. It summarizes the parameters of the operation and offers the administrators the options of performing or cancelling the task. This operation will block all operations due to the backend training, so it is important that it remains opened.

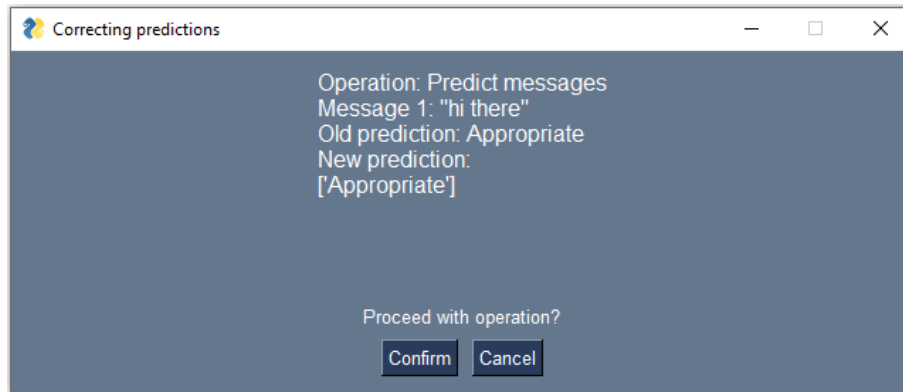


Figure 86 Confirmation window for correcting a prediction

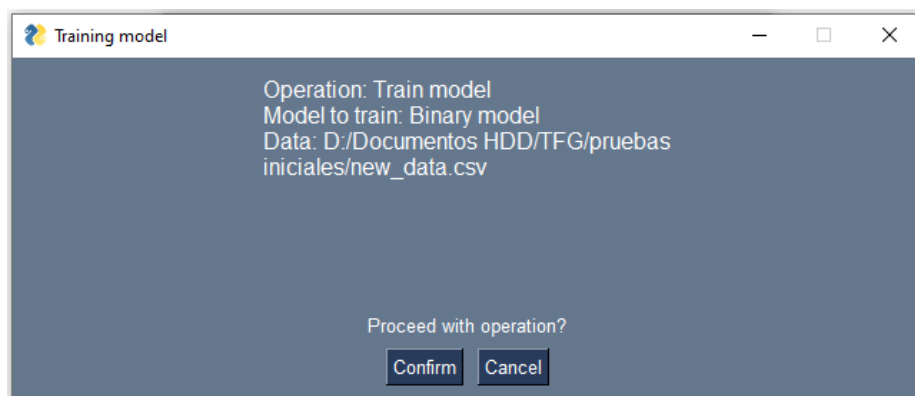


Figure 87 Confirmation window for training a model

6.6 Test plan technical specification

In this subsection the test plan discussed in Test plan specification will be developed to be implemented later in the implementation phase.

All the tests will be executed in a local machine with the following characteristics:

- OS: Windows 10 Education.
- Motherboard: AMD Ryzen A520M-A Prime AM4.
- SSD: Kingston SSD SATA3 V300 120GB.
- RAM: 16 GB DDR4.
- CPU: AMD Ryzen 5 3500X 3.59 GHz. Six cores.

6.6.1 Unit testing

We will discuss the unit tests for every use case and scenario described in section 5.8.5.

6.6.1.1 Change classification method

<i>Use case 1: Change classification method</i>	
Test 1.1	Expected Result
The user changes the method to any of the two possible methods	The controller stores the new in the <code>classifier</code> attribute
Test 1.2	Expected Result
The user changes the method to a not contemplated method	The user interface only offers Binary and Itemized method through a combo box, which is read-only. Also, a default value if asserted to prevent blank method The controller itself validates that the value is correct

Figure 88 Change classification method unit test

6.6.1.2 Detect inappropriate messages

<i>Use case 2: Detect inappropriate messages</i>	
Test 2.1	Expected Result
The user tries to classify a blank message with no file uploaded	The user interface disables the option Classify whenever a message is not typed, and a file is not uploaded The controller checks for each mechanism the emptiness of the message. If it is empty an exception is thrown
Test 2.2	Expected Result
The user tries to classify a message that exceeds maximum message length (500 characters)	The text area displayed allows only to write until 500 characters The controller validates that the message is at most 500-character long. If it is longer, an exception is thrown
Test 2.3	Expected Result
The user tries to upload a file with wrong extension	The Submit messages file option allows only to upload .csv files The controller validates that the extension is correct. If it is not, an exception is thrown
Test 2.4	Expected Result
The user tries to upload with unexpected data	The controller validates the number of columns of the file, which should contain only one The controller validates for every row of the file that the message is a valid character string, that it is not blank and that it does not exceed maximum possible length For any reason above not being fulfilled, an exception is thrown for each message
Test 2.5	Expected Result
The user types a message and tries to upload a file with messages	The user interface disables the Submit messages file option whenever any text is input inside the text area
Test 2.6	Expected Result

The user uploads a file and tries to type a message	The user interface disabled the text area whenever a file is uploaded
Test 2.7	Expected Result
The user types a message with no file uploaded and classifies the message	The controller computes the prediction The user interface shows the prediction
Test 2.8	Expected Result
The user uploads a correct file with valid data and with no typed message	The controller predicts for each message a prediction as the one described before

Figure 89 Detect inappropriate messages unit test

6.6.1.3 Save results to file

Use case 3: Save results to file	
Test 3.1	Expected Result
The user tries to save results to .csv file with no classification performed	The user interface disables the Save results to .csv file option until a classification is performed The controller raises an exception
Test 3.2	Expected Result
The user saves results to .csv of a performed classification to a file	The controller creates a file inside the destination folder, and inserts the last prediction, dumping the message and the obtained prediction For several predictions, each line of the output file will contain the message and the obtained prediction
Test 3.3	Expected Result
The user tries to save results to .txt file with no classification performed	The user interface disables the Save results to .txt file option until a classification is performed The controller raises an exception
Test 3.4	Expected Result
The user saves results to .txt of a performed classification to a file	The controller creates a file inside the destination folder, and inserts the last prediction, dumping the message and the obtained prediction For several predictions, each line of the output file will contain the message and the obtained prediction

Figure 90 Save results to file unit test

6.6.1.4 Correct predictions

Use case 4: Correct predictions	
Test 4.1	Expected Result
The administrator tries to correct a prediction with no classification performed	The user interface disables the option until a classification is performed The controller checks if the <code>last_predictions</code> attribute is empty
Test 4.2	Expected Result
The administrator introduces a correction with wrong values or number of values	The user interface offers combo boxes that will only display the possible values for a prediction. They also have a default value, so the blank prediction is not allowed The controller raises an exception

Test 4.3	Expected Result
The administrator introduces a correct prediction	The user interface shows a message to the user indicating the success of the operation
Test 4.4	Expected Result
The administrator cancels the operation of correcting the predictions	The user interface hides the combo boxes for correcting a prediction
Test 4.5	Expected Result
A non-administrator user tries to correct the prediction	The user interface hides this option for non-administrator users The controller checks if the user is an administrator

Figure 91 Correct predictions unit test

6.6.1.5 Log in as administrator

<u>Use case 5: Log in as administrator</u>	
Test 5.1	Expected Result
The administrator tries to access the Log in as administrator option	The user interface hides this option for administrators The controller checks that the user is not already logged in
Test 5.2	Expected Result
The user selects the option Log in as administrator	The Authentication Window is created and run
Test 5.3	Expected Result
The user tries to introduce blank username or password	The user interface disables the Submit option until any text is written inside the username and the password The controller validates if any of the data is blank. If it is, the controller raises an exception
Test 5.4	Expected Result
The user tries to introduce too long username (20 characters) or password (30 characters)	The controller checks that the username and password do not exceed the possible maximum length (20 and 30 characters respectively). If any does, the controller throws an exception
Test 5.5	Expected Result
The user tries to introduce non-alphanumeric characters for the username	The controller checks that the username is only alphanumeric. If it is not, the controller throws an exception
Test 5.6	Expected Result
The user introduces valid username and password but there is not match in the database	The authentication module returns a negative result (<code>false</code>) because the hashed password and the database password do not match The user interface displays a message for the user
Test 5.7	Expected Result
The user introduces valid and correct username and password	The authentication module returns a positive result (<code>true</code>) The Authentication Window is closed The attribute <code>authenticated</code> in the controller is set to <code>true</code>
Test 5.8	Expected Result
The user cancels the Authenticate operation	The Authentication Window is closed The attribute <code>authenticated</code> remain as <code>false</code>

Figure 92 Log in as administrator unit test

6.6.1.6 Train models

Use case 6: Train models	
Test 6.1	Expected Result
The non-administrator user tries to train a model	The user interface hides this option for the non-administrator user The controller checks if the user is logged in
Test 6.2	Expected Result
The administrator selects the Train model option	The Train Window is created and run
Test 6.3	Expected Result
The administrator tries to upload an invalid file	The user interface only expects .csv files to be uploaded The controller checks the extension of the file. If it is incorrect, the controller raises an exception
Test 6.4	Expected Result
The administrator uploads a file with blank data or wrong prediction values	The classifier checks the validity of the data. If it is not valid, an exception is thrown for each of the rows
Test 6.5	Expected Result
The administrator uploads a file with wrong number of pieces of data	The classifier checks the number of columns of the file. If it is incorrect, an exception is thrown
Test 6.6	Expected Result
The administrator uploads a valid file with valid data and trains the model	The user interface displays the success of the operation
Test 6.7	Expected Result
The administrator cancels the operation	The Train Window is closed

Figure 93 Train models unit test

6.6.2 Integration tests

These tests are covered inside the Unit testing section, that also verifies correct relation between components.

6.6.3 Usability tests

The usability tests will cover the overall satisfaction of the client when it comes to using the application. A third-party user has also been subdued to the system use.

6.6.3.1 User profile ranking

This survey measures the level of experience of a user. It is important than the distinct levels of expertise are able to understand the system scope and what it can do.

How often do you use a computer?
<ol style="list-style-type: none"> 1. Everyday 2. Several times a week 3. Occasionally 4. Hardly ever 5. I have never used a computer
What is your main activity using a computer?
<ol style="list-style-type: none"> 1. It is part of my job or occupation 2. Mainly for free time 3. Using office software 4. Reading news and/or emails 5. I use it for nearly everything
Have you ever used a similar software?
<ol style="list-style-type: none"> 1. Yes, I have 2. No, although I have used software that perform similar tasks 3. No
What do you look forward the most in a program?
<ol style="list-style-type: none"> 1. To be easy to use 2. To perform a lot of tasks 3. To be fast 4. To have a nice interface 5. To be transparent on the operations it performs

Figure 94 User profile ranking survey

6.6.3.2 Guided activities

This survey looks to reflect the advantages and disadvantages of our application, through the main features of the system.

Writing a message and predicting it
<i>Things I liked:</i>
<i>Things I would like to be improved:</i>
Uploading a file with messages and predicting them
<i>Things I liked:</i>
<i>Things I would like to be improved:</i>
Obtaining a file with the computed predictions
<i>Things I liked:</i>
<i>Things I would like to be improved:</i>
Correcting the prediction of a computed classification (Administrators only)
<i>Things I liked:</i>

Things I would like to be improved:
Uploading a file to train a model (Administrators only)
Things I liked:
Things I would like to be improved:

Figure 95 Guided activities survey

6.6.3.3 Quick questions about the application

Ease of use	Always	Most of the time	Occasionally	Never
Do you know where you are inside the application?				
Is there any help when using the application if any doubts arise?				
Do you consider the application to be easy to use?				
Functionality	Always	Most of the time	Occasionally	Never
Does the prediction suit the messages introduced?				
Does the model reflect a better prediction after correcting or training the model?				
Is the files' data format intuitive and suitable?				
Does every task work as expected?				
If any task fails, do you find the error messages descriptive enough?				
Is the response time of the application adequate?				
Interface quality				
Graphic aspects	Very adequate	Adequate	Little adequate	Not adequate
The font size and type is				
Used colours are				
Interface design	Yes	No	Sometimes	
Is the interface easy to use?				
Is the windows design clear?				
Do you think that the application is well-structured?				
Are messages describing thoroughly the specific situation?				
Observations				
The user may detail some experience not covered in the survey				

Figure 96 Quick questions about the application survey

6.6.3.4 Tester survey

This survey is filled by the tester as a conclusion of the responses collected in all the previous surveys. Last column shows decisions taken from the survey results.

Observed aspect	Notes	Possible solutions
<i>The user handles the task in a fast way</i>		
<i>Minor errors</i>		
<i>Major errors</i>		
<i>The predictions are fitting to the true ones</i>		
<i>Files' data format is intuitive to the user</i>		
<i>Output files show content clearly and plainly</i>		
<i>User interface is suitable</i>		

Figure 97 Tester survey

6.6.4 Accessibility tests

Accessibility tests are rarer in desktop applications. All standards and guides are intended for web applications. However, in section 9.3 a customized checklist cherrypicked from [39] will be complimented.

6.6.5 Performance tests

These are the proposed performance tests to benchmark the system.

Test	Workload	Result
<i>1.1 Initial training of binary model</i>	Binary dataset [40]	
<i>1.2 Initial training of multilabel model</i>	Multilabel dataset [41]	
<i>1.3 Predict messages (binary)</i>	10000	
<i>1.4 Predict messages (multilabel)</i>	10000	
<i>1.5 Predict messages from file (binary)</i>	10000	

1.6 Predict messages from file (multilabel)	10000	
1.7 Saving predictions to file .csv	10000	
1.8 Saving predictions to file .txt	10000	

Figure 98 Performance tests

Chapter 7. System implementation

In this chapter the implementation of the system is discussed, along the technologies and tools used, standards and guidelines followed, and problems found during the implementation.

7.1 Programming languages

This section is a list of the programming languages and libraries used in this project accompanied by brief descriptions of each of them.

7.1.1 Python

Python has been used for the totality of the implementation of this project, more specifically version 3.10 Python [42]. It is a high-level programming language widely spread in machine learning applications. It is flexible in its programming paradigm, and it is easy to learn. One of the advantages of Python is the large quantity of libraries that implement machine learning modules, some of which were used in the development of the system and will be described in the following subchapters.

In the last decade, usage of Python has grown exponentially. Some causes are that it is easy to learn and read, the expanding and supportive community, has a large number of well-documented libraries and it is suitable to be used in data science and analytics [43].

7.1.1.1 Scikit Learn

This Python library is commonly used within machine learning projects [32]. It is useful because of its soft learning curve, its simple syntax and fast processing. It has been used in order to implement both binary and itemized models.

Scikit Learn has the key advantage of being easy to understand to the unexperienced developer. The library comes with a wide documentation and supportive community. This is the reason for having taken Scikit Learn as the main modelling option in this project. However, its main drawback is the lack of deep learning modules. The community tried to implement a deep learning Scikit-Learn-approached module that integrated with Keras and PyTorch, but that project was abandoned and has no support [44].

7.1.1.2 Pytorch

Pytorch is a widely used Python library that supports deep learning systems. It is more complex than Scikit Learn, but offers the possibility of constructing artificial neural networks, which are not possible in Scikit Learn.

Along TensorFlow [34] and Keras [45], they are the most common deep learning implementation libraries of the market. These libraries allow developing scalable deep learning systems since the developer may construct a simple baseline model which will gradually expand into a more complex and capable model.

7.1.1.3 Transformers

Transformers is a Python library specialized in Transformers deep learning models. It offers several specialized Transformers models, and also offers base models which may be fine-tuned to a specific task. Several BERT models are available in this library.

Transformer deep learning models have altered the machine learning and deep learning fields. They have more computing capability due to their ability to process the input as whole instead of sequentially, and to keep the positional information of the inputs.

7.1.1.4 PySimpleGUI

This library is the one chosen for implementing the user interface. It encapsulates the implementation of tkinter [46], Qt [47], Remi [48] and WxPython [49], which are common Python libraries. This library is based on list nesting in order to display the desired layout.

7.2 Tools and programs used for the development of the system

7.2.1 Visual Studio Code

Visual Studio Code is the IDE option to develop the application [50]. It is language-independent and possesses a large quantity of plugins that make easier the development. Moreover, this IDE integrates nicely with the GIT version control system.

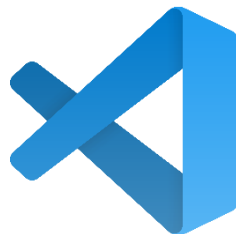


Figure 99 Visual Studio Code Logo

7.2.2 Git

Git is an open-source version control software developed by Linus Torvalds in 2005 [51]. It is the most common version control system in software development, because its better performance and capability in comparison to the other alternatives.



Figure 100 Git Logo

7.3 System creation

The first goal to be achieved inside the implementation of the system was to obtain two functional models that classified messages as appropriate and inappropriate. To make a difference between the two, the idea was to create a more “straight-forward” model that classified the message as one of the mentioned categories, and a more specific model which would detail why a message is inappropriate.

7.3.1 Binary model

The binary model was the easiest one in terms of implementation. When it came to selecting a fitting dataset, the number of alternatives was massive, and so there were a lot of suitable datasets for this system. However, this vastness of datasets became a drawback, which will be discussed later in the Found issues subchapter. The dataset chosen has near 60000 classifications made from downloaded tweets [40].

Decided the dataset, the next task was to choose the most adequate library to implement the classifier. The selection was Scikit Learn by recommendation of the tutor. Given its easy-to-use approach a first model was quickly implemented.

Scikit Learn offers a great number of models to be trained. These are the models that were at least proven to fit the binary model:

- LogisticRegression.
- Perceptron.
- SGDClassifier.

The data obtained from the dataset should be transformed in some way to better fit the computer processing capabilities. Most of the machine learning models accept to different sets of data: the features and the targets. Feature is any information that we provide the model classify it, for instance the message. There could be other features calculated from the former,

like length of the message, number of occurrences of certain words, etc. Targets are the classification itself (if it is appropriate or not), and in this case we have only one.

Targets come in several formats depending on the dataset. There are datasets that define the target with words like “appropriate” or “toxic”. In the case of our dataset, the target was already established as a number (0 for appropriate, 1 for inappropriate). This is a good improvement, since computers manage numbers faster than words.

Features normally go into a more thorough process. This transformation was an issue that will be described in detailed in the Found issues section. The final decision of the features transformation was to use a bag-of-words approach, which basically means to create a $N \times M$ matrix, being N the number of samples of the dataset and M the number of total words compiled of message. Inside each cell of the matrix, the number of occurrences of the m word in the n message is stored [52].

There is also a step in which the dataset is split in a training subset and a validation subset. This is done to prevent the model from overfitting, that means that the model precisely classifies the provided data but is prone-to-error to unseen data, which is a symptom of a bad performance.

The next step after establishing the training process of the dataset was to perform natural language processing tasks. A substantial amount of time was dedicated to this phase. This step is important because we are trimming the amount of information that the computer has to process. There is a lot of layers in a text which does not supply any semantic information, like punctuation and stop words.

Contraire to general belief, the more that you refined this processing did not result in better results. There is an optimal point on which certain NLP tasks would lead to the maximum performance of the model.

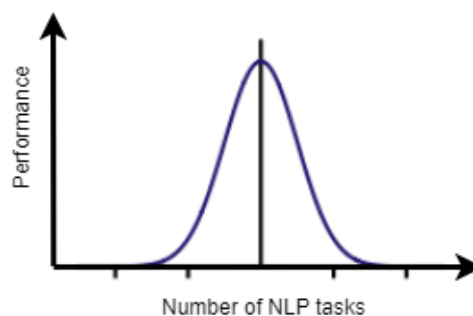


Figure 101 Natural language processing graph

The natural language processing tasks performed include:

- Accents removal.
- Punctuation removal.
- Stop words removal.
- Hashtags, usernames, and hyperlinks processing.
- Undo contractions.

- Regular expression fulfilment.
- Tokenization.
- Lemmatization and stemming.

Then, it became a process of refinement of the already established model, like changing hyperparameters inside the LogisticRegression and SGD classifier, the ratio of training/validation data, the transformation process, etc. Finally, the SGD classifier was the selection for the binary model. The statistic backing this decision are commented in section 8.2.1.

7.3.2 Itemized model

This was the part which consumed most of the time of the system implementation, and it brought most of the problems and delays of the project fulfilment.

The dataset research was not difficult. The Jigsaw Toxic Comment Classification Challenge posted this competition in Kaggle in 2018 including this dataset. It contained a training set of 150000 entries and a validation set of 40000 [41]. There are some other datasets that are offered in the web with itemized hate-speech-related classifications, but most of them were a sort of modification from the Jigsaw one.

The main concern with the library selection was that the intended approach was to create an artificial neural network that could perform this itemized classification. Since native Scikit Learn does not handle deep learning, Pytorch was the option for implementing it. However, due to issues, which are discussed below, related to the neural network poor performance and weak computational capability, the neural network model was discarded, and the decision was to go back to the Scikit Learn library.

The models used had to be more complex, given the more complexity of this system. This sort of classification tasks, which are called multilabel classification tasks, are not the main goal of Scikit Learn, which is more focused into different tasks like binary classification, regression, and clustering. However, it offers a MultiOutputClassifier [53] that receives a classifier and computes a multioutput (multilabel) classification. This was quite suitable for our system, since a binary model based on a LogisticRegression classifier had been already implemented.

Feature transformation on this model follow the same pattern and problems than the binary one. The solution was to use a slightly different approach to bag-of-words, which was use TF-IDF, which stands for Term Frequency times Inverse Document Frequency [54]. This solution changes occurrences to frequencies and penalizes words that appear too much in a lot of documents, like connectors.

Finally, some hyperparameter refinement was performed to achieve better results. The final decide model was the MultiOutputClassifier assembled with the SGD classifier. This decision is reasoned in section 8.2.2.

7.3.3 Application and interface

The ending part of the process was to create an application that could encapsulate the prediction capability of both models. Some features were added to this application to offer more utility, like saving the predictions to an output file, uploading a file with several messages, and allowing an administrator to correct the models.

The key principles upon which the system was created was high-cohesive and low-coupled classes, clear separation between interface, logic and persistence levels, and a strategy pattern applied to the system. The strategy pattern allows a class to perform the same operation in different ways, which is the case in our system. The main advantage is that for other model to be added, it would only be needed to implement the Classifier interface with the necessary methods and probably a Prediction class to pass it to the interface.

The interface was the last part of the system implementation. Although the first idea was to create a simple interface with no sophisticated animations or transitions, the inexperience and PySimpleGUI illegibility took large roles when it came to the interface implementation.

7.3.4 Last modifications

After results listed in section 9.2 of the usability tests done, some modifications were added to the system to better suit users' demands.

- An option that allows to save the results of a prediction in a user-friendly format into a .txt file.
- Both .txt and .csv file save processes now allow the user to select the destination of the file.
- An option of refreshing the current prediction. Since the only way to perform the same prediction is to press the Clear All option and rewrite it, this option allows to skip these steps. Also, it is useful after the correction of a prediction to see the alterations.
- Two options, both in the main window next to the Submit messages file option and in the training window to offer some help on the files' format. Also, a tooltip is shown with an example of a file row for each of the models.
- Some of the text within the window has been augmented for better readability.
- Some elements have been relocated for better display.

7.3.5 Found issues

During the whole process of the implementation, several problems were encountered, which will be analysed in the following subsections.

7.3.5.1 Subjectivity of the term “inappropriate”

7.3.5.1.1 Description

This problem is carried throughout the entirety of this project. The term is defined as «something which is not suitable for a particular situation». Then, the problem arises on this situation. Something can be inappropriate in a situation but be appropriate for another one.

In text classification this term becomes even more misleading. Although it is true that there are common boundaries upon which everyone would agree like racial or misogynistic slurs, these boundaries become dimmer when the text has not clear harmful intentions.

For instance, consider an angry review of a film. Would the sentence «This film is s***» be an inappropriate sentence? Some may argue that the final word would determine the classification. However, the sentence could also say «This film is trash», and in that case the word «trash» is not considered a bad word in English, but the critique is the same. Moreover, the sentence could be «This film is so bad that everyone who worked doing it should be fired». In this case, the sentence does not contain a single word which could be individually understood as inappropriate. The whole semantic confers offensiveness to the sentence, and in this case, it could be said that is even more offensive than the first two sentences. There could be major agreement in the first example, but the last one is not so obvious.

This unprecise term leads to a whole range of available datasets on the Internet. Some keywords used for searching them were «hate speech», «offensive comments», «toxic comments», etc. For the most part, these datasets are put to test to several individuals that classify them as offensive/inoffensive or toxic/non-toxic to later do an arithmetic mean. And this is again the issue, that the different opinions lead to different results.

For the itemized model it becomes even more exacerbated, due to several targets being discussed instead of only one. This will generate other issue that is commented below.

7.3.5.1.2 Proposed solution

The unescapable nature of the term leads to state a formal definition in the context of this project. «Inappropriate» is a property socially conferred to whichever message that exhibits elements of toxicity, obscenity, threat, insult, and/or hate speech within its meaning. In essence, a message is inappropriate if most of the people can classify it in at least one of these categories.

Both datasets attach to this definition. For the binary dataset, all messages were analysed by several people classifying it as toxic or non-toxic. The itemized dataset is classified in terms of toxicity, severe toxicity, obscenity, threat, insult, and identity hate.

7.3.5.2 Multilabel unbalanced dataset

7.3.5.2.1 Description

The Jigsaw Challenge dataset is the most used dataset in toxic comment classification tasks. One of its main characteristics is that data is not balanced at all. Around 89% of the comments are classified as not harmful, which is having all targets to 0. From the 11% remaining, the most common label is the “toxic” one and the rarest is the “threat” label. This issue is known as dataset unbalance.

Let’s see an example to demonstrate why this is an issue. Imagine a model that classifies an image as a dog or as a cat. We want to provide the model with similar number of samples of both classes so it can “nourish” equally from them. If we had a training dataset that had 90% of dog samples and 10% of cat samples, and trained it, the classifier would correctly identify most of the dogs, but for an unseen image of a cat, it would probably classify it as a dog too, so this model performs poorly.

There are mechanisms to overcome this problem, like oversampling, which consist of creating new artificial data mixing features, in this case sentences, between them and generating more samples. However, a user in the Kaggle competition pointed out that this unbalance is necessary for the model because the unbalance provides information to the model about the frequency of toxic comments. For instance, you want the model to know that threat messages are less common than toxic ones, and so be more confident when predicting it [55].

7.3.5.2.2 Proposed solution

The only solution was to handle this unbalanced dataset as an unavoidable issue.

7.3.5.3 Lack of computational power

7.3.5.3.1 Description

The implementation of the system was developed in two different computers:

- A desktop PC with Windows 10, 16GB of DDR4 RAM, a Kingston V300 SSD of 120GB, an AMD Ryzen 5 3500X CPU and a NVIDIA GeForce GTX 760 graphic card.
- A laptop with Windows 10, 8GB of DDR4 RAM, an Intel i5-6300HQ CPU and a NVIDIA GeForce GTX 950M graphic card.

These computers are well-functional, and they are 8 and 5 years old respectively. For the 99% of tasks, they work perfectly fine. However, machine learning and deep learning are resource-demanding processes that require great computational capability.

The training phase of a machine learning model is usually done with a graphic card, given it is fast when working with matrices. To prepare the graphic card for this tasks, CUDA (Compute Unified Device Architecture) is mandatory [56]. The GPU of the first computer is not supported

by CUDA, and so it is not able to perform training operations. The laptop GPU is indeed supported by CUDA, but it is not a powerful graphic card.

Artificial neural networks are even more resource-demanding since they must perform back propagation algorithms. The options are to train it in the desktop's CPU or to do it in the laptop's GPU. The CPU takes around last 30 times longer than the laptop's GPU, and so it is not feasible to do it. In the laptop, times were more reasonable (20 hours per iteration), but the performance was not great. This issue is commented below.

7.3.5.3.2 Proposed solution

Since neural networks require great resources, the alternative was not to use a neural network, but a simpler model implemented in Scikit-Learn.

7.3.5.4 *Neural networks do not perform well*

7.3.5.4.1 Description

Several neural networks configurations written in Pytorch were attempted. These neural networks did not achieve any substantial performance, being for some cases extremely poor and achieving accuracy, precision and recall of 0.

Speaking in technical terms, the activation function for the output of the neural network was a sigmoid function with a 0.5 threshold. The observations made established that any message, regardless of its content, would have a sigmoid activation of less than 0.1, so every prediction would be a 0 (a positive prediction). If the threshold was shifted to be lower to, for instance, 0.1, it would make a "rebound effect" and classify most of the messages with a negative prediction for every target.

Neural networks poor performance is cause of several factors. The first one is shallow knowledge about neural networks. To be able to construct a working artificial neural network, it is key to understand activation functions, embeddings, attention layers or even self-contained structures that could be a layer in our neural networks, like GRU or LSTM models [57], [58]. The second factor is the one described above, which is the demanding computational power to train a neural network. They require large amounts of iterations adjust to the suggested task. The third one is having a small period of time available to run the training.

Nevertheless, a final neural network structure was attempted. This neural network was found in a GitHub repository of the user "jonad" [59].

The results of using this structure were relatively good. Despite of this sudden improvement, it did not perform remarkably better than the MultiOutputClassifier with LogisticRegression of Scikit Learn, so the final choice was to stick to the later, given it is way faster to train and predict.

7.3.5.4.2 Proposed solution

As commented before, the solution was to get rid of the neural network idea and make a Scikit Learn multioutput model.

7.3.5.5 BERT model does not perform well

7.3.5.5.1 Description

This may be seen a subset of the problem above. A BERT model was used for some neural networks' structures and did not perform well. The possible causes were commented previously.

7.3.5.5.2 Proposed solution

The alternative was not to use the BERT model, but a simpler model implemented in Scikit-Learn.

7.3.5.6 Data transformation

7.3.5.6.1 Description

As we have commented in the Binary model section, the transformation picked was bag-of-words and the TF-IDF approach. These methods are resource-cheap and fast, but they have a huge disadvantage. The matrix computed for both methods measure the occurrence or frequency of the words in the samples but lose all meaning. The last example shown in the description of the Subjectivity of the term "inappropriate" issue spoke about how there are sentences that are offensive without using bad vocabulary, so there is a loss of semantic with those approaches.

A common alternative for transforming text is word embeddings. A word embedding is essentially a vector that represents the word. If we saw a visual representation of these vectors, assuming they are three-dimensional, we could see that similar words like "king" and "queen" are close in the space, but words like "videogame" are farther. Word embeddings come from pretrained machine learning models that have generated them. Hundreds of thousands of words are represented by vectors of dimension hundred, two hundred or even more dimensions. Popular word embeddings are word2vec [60], glove [61] and fasttext [62]. They keep the meaning of the words, but they more resource-intensive and last longer to compute for the specific task they are used in.

Word embeddings were tried in both Scikit Learn and Pytorch implementations. Scikit Learn did compute decent results, but the ones with bag-of-words or TF-IDF were better, and Pytorch neural networks suffered from the problem of performance described Neural networks do not perform well.

7.3.5.6.2 Proposed solution

The data transformation phase remained in the bag-of-words and TF-IDF methods.

7.3.5.7 Best metric for measuring overall performance

7.3.5.7.1 Description

When evaluating a machine learning model, it is common to rely on accuracy as the best indicative of overall performance. Despite the fact it is a good metric, depending on and improving exclusively this metrics disregarding all the other ones is counterproductive. High accuracy is not always a sign of a good model.

To understand the issue, we will see how metrics are calculated for a binary classification. A confusion matrix summarizes the results of a classification of a set of messages.

		Predicted Values	
		Negative (0)	Positive (1)
Actual Values	Negative (0)	TN	FP
	Positive (1)	FN	TP

Figure 102 Confusion matrix

In each cell a specific value is stored:

- TP: True Positives, are the number of samples predicted as positive that are actually positive.
- FP: False Positives, are the number of samples predicted as positive that are actually negative.
- FN: False Negatives, are the number of samples predicted as negative that are actually positive.
- TN: True Negatives, are the number of samples predicted as negative that are actually negative.

We will see an example explaining this matter. Picture the model mentioned before that can classify images as being a cat or being a dog. We now consider that a new set of unseen data is presented to the model. This data contains 9 images of a dog and 1 of a cat. This model performs badly and recognizes every image as a dog. This is the resulting confusion matrix:

		Predicted Values	
		Dog	Cat
Actual Values	Dog	9	0
	Cat	1	0

Figure 103 Dog/cat confusion matrix

The accuracy is calculated as $\frac{TP+TN}{TP+TN+FP+FN}$, so it would be an accuracy of 90%. We could say that the model is good because it achieves an accuracy of 90% for unseen data, but we can see that the real issue is that this specific set of data contained mostly dogs, and so the high value of the metric.

In addition to this, accuracy is not suitable for multilabel classification. Our itemized model classifies a message into six non-exclusive binary targets. Now, imagine a classification of a message outputs that it is toxic, obscene, and insulting. We could represent it in a vector as $[1,0,1,0,1,0]$. When we see the true value of prediction of the message we get $[1,1,1,0,1,0]$, that is toxic, severe toxic, obscene, and insulting. The accuracy metrics would consider this prediction as wrong, but we can see that it is only partially wrong, given that it was well classified except for the severe toxic target.

Then, it is mandatory to focus on other metrics to measure the performance of a model.

7.3.5.7.2 Proposed solution

For the binary model, accuracy is evaluated along other metrics were considered like Precision, Recall and F1 score [63]. For the multilabel model accuracy is not as valuable, so the focus is shifted towards metrics like F1 score, Precision, Recalls and Hamming Loss [64].

7.3.6 Detailed class description

Inside attached file, we can see a `doc/` folder, and a `pydoc/` subfolder which contains the documentation generated with Pydoc [65]. We should open the `src.html` file to be able to browse through all the files.

Chapter 8. Evaluation of alternatives

This chapter exposes some solutions and statistics found on the web about toxic comment classification. Later, the different alternatives tried on both models are discussed along the metrics to evaluate them.

8.1 State of art solutions and statistics

These solutions were found mostly the research phase, and some of them throughout the whole process of fulfilling of this project. We will divide them into the binary models and the multilabel/itemized ones.

8.1.1 Binary models

A table showing the different solutions for binary models is presented. Bold values are the highest among the corresponding metric. The next list shows the paper names:

1. Detecting toxic behaviour in social media and online news [66].
2. Toxicity Detection on Bengali Social Media Comments using Supervised Models [3].
3. MC-BERT4HATE: Hate Speech Detection using Multi-channel BERT for Different Languages and Translations [5].
4. Deep Learning for Hate Speech Detection in social media [67].
5. Overview of MEX-A3T at IberLEF 2019: Authorship and aggressiveness analysis in Mexican Spanish tweets [4].
6. Kaggle's user doing the classification of Toxic tweets [68].

Paper	Model	Precision	Accuracy	Recall	AUC	F1 Macro	F1
1	GRU + Attention	0.86	0.84	0.53			0.66
	GRU + Capsule	0.73	0.84	0.73			0.73
	MultiCNN	0.86	0.83	0.49			0.63
	BERT	0.85	0.87	0.66			0.74
	BERT-P	0.85	0.85	0.6			0.71
	Ensemble BERT	0.84	0.87	0.66			0.74
2	Naive Bayes		0.818				
	SVM		0.8473				
	Logistic Regression		0.8522				
	LSTM		0.9413				
	CNN		0.953				

3	MC-BERT fine-tuning (HatEval)	0.769				0.77
	English BERT fine-tuning (HatEval)	0.752				0.75
	Chinese BERT fine-tuning (HatEval)	0.7				0.69
	Multilingual BERT fine-tuning (HatEval)	0.755				0.75
	MC-BERT fine-tuning (GemEval)	0.801				0.76
	English BERT fine-tuning (GemEval)	0.798				0.77
	Chinese BERT fine-tuning (GemEval)	0.76				
	Multilingual BERT fine-tuning (GemEval)	0.779				0.74
	MC-BERT fine-tuning (HaSpeede)	0.8				0.78
	English BERT fine-tuning (HaSpeede)	0.798				0.77
	Chinese BERT fine-tuning (HaSpeede)	0.799				0.78
	Multilingual BERT fine-tuning (HaSpeede)	0.822				0.8
	4	CNN + LSTM	0.925			
5	CNN + LSTM, Multilayer perceptron (PRLHT)	0.7			0.63	
	SVM	0.68			0.59	
	Naive Bayes	0.69			0.61	
	SVM	0.71			0.63	

	SVM + Multilayer perceptron		0.73			0.65	
	CNN; LSTM; GRU VRAIN		0.61			0.51	
6	BiGRU		0.925	0.925			0.93

Figure 104 Statistics from state-of-art binary models

A general issue with machine learning solutions on the Internet is that the most popular metric for evaluating the models is the accuracy. Having a good accuracy, although is not a bad performance, is not guaranteed of a good one either as isolated evaluation. This concern is discussed in the Best metric for measuring overall performance subsection.

We can observe the different values that several proposed models of the accuracy. Some studies also provide the recall and F1-score metrics. An interesting aftermath can be inferred: deep learning methods, which are more resource-expensive, do not imply better results than the machine learning methods. We can see that for paper **1** a CNN [69] with an attention layer is proposed and an accuracy of 84% is achieved. In paper **2**, a Logistic Regression model reaches an 85.22% of accuracy. Bear in mind once again that accuracy does not tell the whole story. As we can see, the CNN of paper **1** has also a good precision (86%), and it could be possible that the Logistic Regression model had a poor precision value. We cannot establish a formal comparison with lack of information. We can only partially evaluate the models on common metrics.

One interesting case is paper ¡Error! No se encuentra el origen de la referencia., which uses a bidirectional gated recurrent unit deep learning model [70] to perform the classification of the same dataset used in our binary model [40]. It has also the best performing recall F1-score of the solutions.

However, the proposed model in this project firmly exceeds those values. We will see the specific values in the Binary models section.

8.1.2 Multilabel models

These are the multilabel models studied. Bold values are the highest among the corresponding metric. The next list shows the paper names:

1. Jigsaw Challenge 2nd place [71].
2. Jigsaw Challenge 3rd place [72].
3. Jigsaw Challenge 5th place [73].
4. Application of Recurrent Neural Networks in Toxic Comment Classification [74].

Paper	Method	Precision	Accuracy	Recall	AUC	F1 Macro	F1
-------	--------	-----------	----------	--------	-----	-------------	----

1	RNN + DPCNN + GBM		0.98822				
2	BiLSTM + BiGRU		0.9872				
3	Two level BiGRU		0.9865				
4	Baseline	0.48 - 0.8672		0.08 - 0.7783	0.9797		
	GRU	0 - 0.8277		0 - 0.8079	0.9782		

Figure 105 Statistics from state-of-art multilabel models

Note that all models were trained with the Jigsaw challenge dataset [41]. The first three models were taken from the Jigsaw competition itself. The competition awarded the ranking position in terms of accuracy, so it is the only metric available for those models. Their accuracy is quite high, given that their models are quite complex. Most of the high ranks were populated by teams with expert knowledge of deep learning that dedicated large amounts of time and resources in this competition. They apply advanced mechanisms like TTA [75], dense word embeddings and model assembling. A common characteristic of these models is that the text preprocessing phase was not devoted a lot of time. The main concern for the teams was to create the right neural network structure and try several word embeddings that better fit their specific model.

There is also a fourth solution that was not taken from the Kaggle competition. This solution proposed two models: a baseline model implemented in Keras [45] and a gated recurrent unit [57]. An interesting feature of this paper is that it did not present an accuracy measure. For the metric values presented they vary drastically depending on the target measured. This is probably due to the dataset unbalance, which is discussed in the Multilabel unbalanced dataset issue description. Nevertheless, the value for the AUC [76] is considerably big.

Our multilabel model does not reach those high values. Class imbalance played a major role in this issue, as it is commented in the Multilabel unbalanced dataset issue description. We will see the specific values of our statistic in the 8.2.2 section.

8.2 Attempted alternatives

In this section, the different alternatives attempted during the implementation of both binary and multilabel systems is commented along some statistics to explain the decision of choosing the final models.

8.2.1 Binary models

All tried binary models were implemented in Scikit Learn, since one of the first classifiers implemented performed well, so the decision was to perform alterations to different Scikit Learn models, in the data transformation and the text preprocessing part.

This table shows different alternatives attempted. Bold values are the best in each metric. A legend is presented explaining the table content.

These are the header abbreviations:

- **DT**. Data transformation method.
- **NLP**. Natural language processing tasks.
- **ACC**. Accuracy metric.
- **PREC**. Precision metric.
- **F1**. F1 score metric.

These are the different models:

- **1**. Logistic Regression LBFSG Solver [77] L2 Regularization [24].
- **2**. Logistic Regression Liblinear Solver L2 Regularization.
- **3**. Logistic Regression SAG Solver L2 Regularization.
- **4**. Logistic Regression SAGA Solver L2 Regularization.
- **5**. Perceptron [78] L2 Regularization.
- **6**. Perceptron L1 Regularization.
- **7**. Stochastic Gradient Descent [25], [79] Hinge loss function [26].
- **8**. Stochastic Gradient Descent Modifier Huber loss function.
- **9**. Stochastic Gradient Descent Perceptron loss function.
- **10**. Stochastic Gradient Descent Huber loss function.

The different datasets:

- -1-. Labelled Data [80].
- -2-. Final Balanced Dataset [40].
- -3-. List of bad words [81].

The NLP tasks used:

- TOK. Tokenization [82].
- SR. Stop words Removal.
- PR. Punctuation Removal.
- ULUR. Username, Link and Unicode symbols Removal.
- STEM. Stemming [83].
- CE. Contractions' expansion.
- LOW. Text to lowercase.

The different data transformation methods:

- (1). Bag of words [54], n_gram_range = {1,2} [84].
- (2). Bag of words, n_gram_range = {1,1}.
- (3). Bag of words, n_gram_range = {1,3}.
- (4). TF-IDF [54], n_gram_range = {1,2}.
- (5). TF-IDF, n_gram_range = {1,1}.
- (6). TF-IDF, n_gram_range = {1,3}.

- (7). Bag of words, n_gram_range = {1,2} Binary¹ Strip Accents

Finally, the colours represent different stages of the trial of these alternatives:

- Blue. The first models tried when implementing the binary model.
- Green. Combination of the dataset with some external lexicon.
- Red. Execution of almost every possible combination of dataset, data transformation method and NLP tasks. See **¡Error! No se encuentra el origen de la referencia..**
- Yellow. Best data transformation method and combination of NLP tasks.

Model	Dataset	NLP	DT	ACC	PREC	REC	F1	Time
1	-1-	TOK + SR + PR + ULUR + STEM	(1)	0.80944	0.75054	0.80944	0.80884	0.76s
1	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.95013	0.92024	0.95013	0.94988	15.91s
2	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.95022	0.92043	0.95022	0.94997	12.98s
3	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.95013	0.92039	0.95013	0.94988	25.15s
4	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.94995	0.92015	0.94995	0.9497	34.24s
3	-2-	TOK + SR + PR + ULUR + STEM	(2)	0.94889	0.91749	0.94889	0.94866	16.19s
3	-2-	TOK + SR + PR + ULUR + STEM	(3)	0.94889	0.91885	0.94889	0.94862	39.62s
3	-2-	TOK + SR + PR + ULUR + STEM	(4)	0.93902	0.89902	0.93902	0.93877	12.92s
3	-2-	TOK + SR + PR + ULUR + STEM	(5)	0.94026	0.90587	0.94026	0.93986	11.64s
3	-2-	TOK + SR + PR + ULUR + STEM	(6)	0.93488	0.88867	0.93488	0.93472	14.07s

¹ The binary parameter means that the bag of words does only track if the words appear in each document, disregarding the number of occurrences

5	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.90007	0.93634	0.90007	0.8994	12.5s
6	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.92898	0.87815	0.92898	0.92882	12.83s
7	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.95163	0.92094	0.95163	0.95143	12.73s
8	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.95189	0.92027	0.95189	0.95172	12.7s
9	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.94264	0.90494	0.94264	0.94241	12.52s
10	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.9259	0.88866	0.9259	0.92501	12.8s
3	-2- -3-	TOK + SR + PR + ULUR + STEM	(1)	0.94995	0.91962	0.94995	0.94971	28.2s
8	-2- -3-	TOK + SR + PR + ULUR + STEM	(1)	0.94484	0.90402	0.94484	0.94476	14.44s
3	-2-	TOK + STEM	(1)	0.94713	0.91568	0.94713	0.94685	80s
3	-2-	TOK + SR + CE + PR + ULUR + STEM	(1)	0.94704	0.91526	0.94704	0.94677	81.19s
3	-2-	TOK + SR + PR + ULUR + STEM	(1)	0.9466	0.91458	0.9466	0.94632	81.4s
3	-2-	TOK + LOW + SR + PR + ULUR + STEM	(7)	0.95321	0.9241	0.95321	0.95302	63.45s
8	-2-	TOK + LOW + SR + PR + ULUR + STEM	(7)	0.95365	0.92291	0.95365	0.95351	62.6s

Figure 106 Binary model alternatives

The first dataset ever tried on the implementation of the binary model was the dataset used in [1]. Although it is intended to be for multiclass classification [85] with three classes, it was transformed to a binary dataset, converting hate speech and offensive language classes to the same class. Also, the first model was a logistic regression with the default solver. This model performed well and was executed instantaneously.

Then, a new balanced dataset [40] was found in Kaggle. This dataset compiled some unbalanced datasets that the author had found throughout his work. As the results were better in every aspect but the time, the first dataset was no longer tested.

After finding this dataset, other logistic regression solvers were attempted, and the best one was the Stochastic Gradient Descent (SAG) solver, which as Scikit Learn shows, is optimal for big sets of data [86]. Regularization was kept to L2 since it's the most aggressive to prevent overfitting. This classifier was tried with different bag of words and TF-IDF combination, being the bag of words with a ngram range of {1,2} the most suitable one.

Other models were tried like the Perceptron and the Stochastic Gradient Descent classifiers. The Stochastic Gradient Descent with the 'modified huber' loss function performed better than any previous model.

Moreover, a test was performed on third-party lexicons to be added as an additional classification feature.

After several trials, an optimal preprocessing and data transformation method was established. In this case, the Stochastic Gradient Descent still performed the best.

The red rows of the table show the best three models obtained in the execution of a file attempting several combinations. This process is described **¡Error! Marcador no definido.¡Error! No se encuentra el origen de la referencia..**

8.2.1.1 Binary model combinations

In a point of the development process of the binary model, a script that executed eighty possible combinations of dataset, preprocessing, and data transformation method. The output of this script is in the file `alltests.log` file attached inside the directory `doc/logs/`.

For the datasets, both [40] and [41] were utilized. Although the latter is a multilabel dataset it was converted to a binary one, by making comments with no toxicity, obscenity, threat, insult, or identity hate the appropriate ones, so the rest are inappropriate.

For the NLP tasks ten different possibilities were offered, and the data transformation methods possibilities were bag-of-words, doc2vec [87], fasttext [62] and glove [61].

As the file contains some abbreviations, they are described here:

- FBD. Final Balanced Dataset [40].
- jigsaw. Jigsaw Dataset.
- clean_text1. TOK + SR + PR + ULUR + STEM.

- clean_text2. Every option was modifiable:
 - no lower: Text is not converted to lowercase.
 - ws: White spaces are kept.
 - no numbers: Numbers are kept.
 - no abbrev exp: Contractions are not expanded.
 - no tw processing: Twitter-related information is kept. Hashtags are not separated into words.
 - sw: Stopwords are kept.
 - no token: Tokenization is not performed.
- bow. Bag of words
- d2v. doc2vec

As to show the statistics would be overwhelming, some conclusive charts are presented.

First, Figure 107 shows the different combination grouped by their data transformation method. We can see that bag of words is the most consistent method in all the combinations. Word embeddings tend to perform better on the [41] dataset.

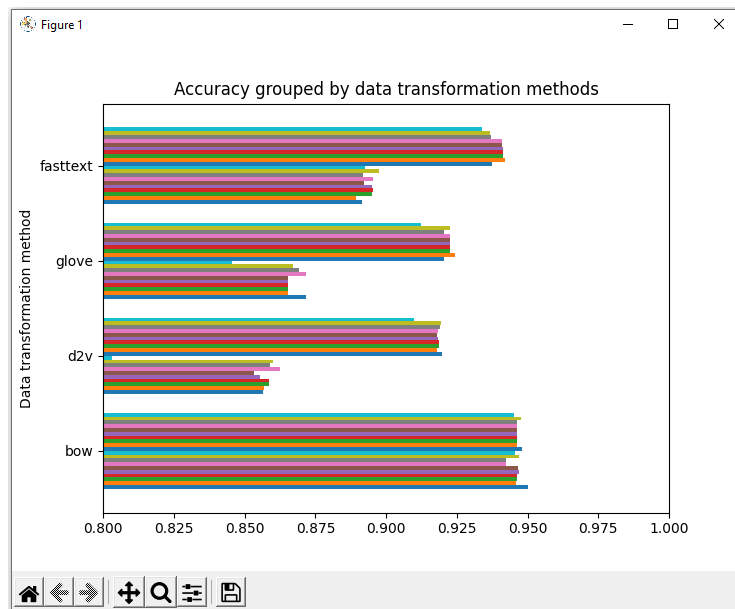


Figure 107 Accuracy grouped by data transformation methods

Next, the results grouped by preprocessing possible combinations is shown in Figure 108. We can see that they hardly change the overall result of the dataset accuracy. A conclusion of this graph could be that NLP tasks had little impact on the accuracy of the models.

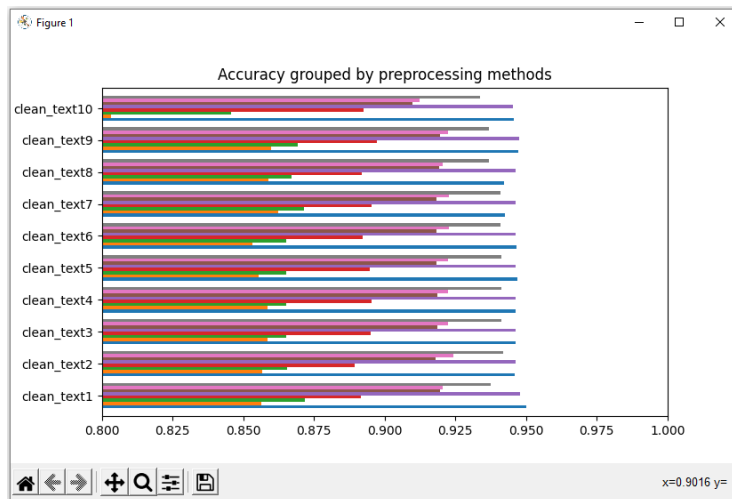
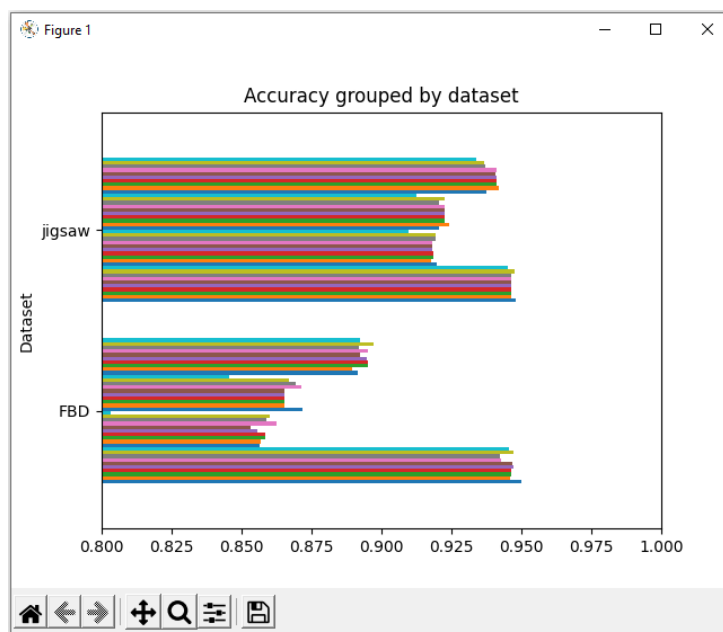


Figure 108 Accuracy grouped byu preprocessing methods

Finally, the results were clustered by the used dataset. At first glance, the Jigsaw binarized datasets models tend to perform more consistently in Figure 109.



However, if we focus on another metric like precision in Figure 110, the opposite happens.

Figure 109 Accuracy grouped by dataset

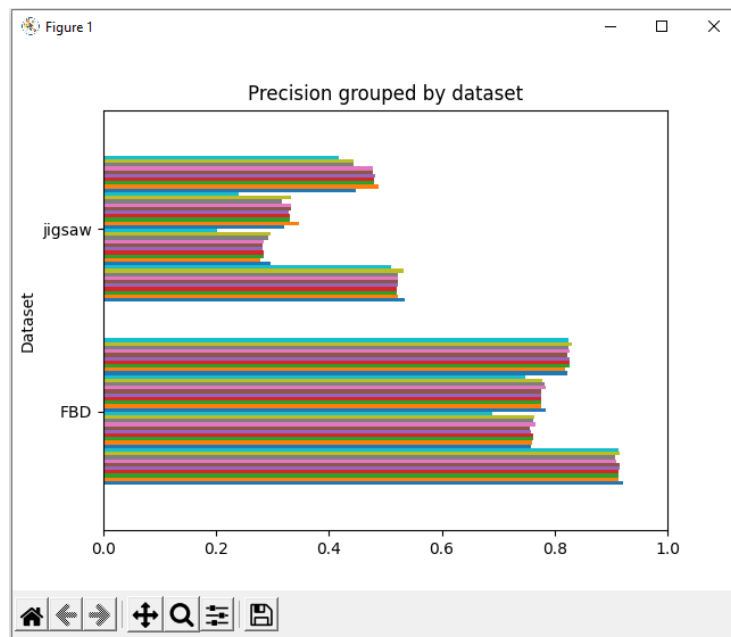


Figure 110 Precision grouped by dataset

8.2.2 Multilabel models

The alternatives of multilabel models were implemented in both Scikit Learn [32] and PyTorch [33]. Figure 111 shows these alternatives. Bold values highlight the highest values on the corresponding metric. A legend is presented in order to understand the information inside:

Header abbreviations:

- **DT.** Data transformation method selected.
- **ACC.** Accuracy metric [63].
- **PREC.** Precision metric.
- **REC.** Recall metric.
- **F1.** F1 Score metric.
- **HL.** Hamming loss metric [64].
- **JS.** Jaccard score metric [88].

These are the different models used:

- **1.** BinaryRelevance [10] + Logistic Regression [6] SAG Solver [77] L2 Regularization [24].
- **2.** ClassifierChain + Logistic Regression SAG Solver L2 Regularization.
- **3.** Label Powerset + Logistic Regression SAG Solver L2 Regularization.
- **4.** OneVSRest [9] + Logistic Regression SAG Solver L2 Regularization.
- **5.** OneVSRest + Stochastic Gradient Descent [25], [26] Modified Huber Loss Function [26].
- **6.** BERT [30], [60] + Dropout [89] of 0.3 + Linear Layer [90].
- **7.** Embedding [60] + Dropout of 0.3 + Linear Layer.
- **8.** Bidirectional LSTM [58] + Attention Layer [91].
- **9.** Multioutput [53] + Logistic Regression SAG Solver L2 Regularization.

- **10.** Multioutput + Stochastic Gradient Descent Modified Huber Loss Function.

The dataset used is [41], which is discussed in section 7.3.2.

The text preprocessing was the same in every model: Tokenization [82] + Text to lowercase + Hashtag split into several words + Stop words removal + Punctuation removal + Username/Links/Unicode symbols removal + Stemming [83].

The data transformation methods:

- (1). Bag of words [52] n gram range (1,2) min_df = 10².
- (2). Bag of words n gram range (1,2) Binary.
- (3). TF-IDF [54] n gram range (1,2) Binary.
- (4). BERT Tokenizer [92].
- (5). Glove [61] Word Embedding (50-dimensional vectors).
- (6). Glove Word Embedding (200-dimensional vectors).
- (7). TF-IDF n gram range (1,1) min_df = 25.

Finally, the colour code is the following:

- Blue. Scikit Learn transformation methods. It has been achieved with a non-native Scikit Learn library called "skmultilearn" [93].
- Green. One VS Rest (OvR) wrapper for binary models.
- Red. PyTorch artificial neural networks.
- Yellow. Scikit Learn Multioutput wrapper for binary models.

Model	DT	ACC	PREC	REC	F1	HL	JS	Time
1	(1)	0.87968	0.40399	0.62574	0.60792	0.02978	0.44181	2097s
2	(1)	0.89567	0.37485	0.58732	0.57076	0.03112	0.41078	2410s
3	(1)	0.89088	0.34861	0.50628	0.55037	0.03053	0.38582	4319s

² This parameter indicates the number of minimum of documents where each word should appear to be taken into account

4	(2)	0,97284	0,31851	0,97284	0,97435	0.03117	0.95356	2179s
4	(3)	0.97393	0.28018	0.97393	0.97305	0.02606	0.95448	436s
5	(2)	0.96955	0.27	0.96955	0.97003	0.03046	0.94927	374s
5	(3)	0.97473	0.26006	0.97473	0.97298	0.02528	0.95489	426s
6	(4)	0	0	0	0	N/M	N/M	80h
7	(5)	0	0	0	0	N/M	N/M	80h
8	(6)	0.91145	0.5	0.28	0.34167	N/M	N/M	80h
9	(7)	0.89454	0.4506	0.63174	0.64417	0.02556	0.48109	265s
10	(7)	0.89757	0.44586	0.61373	0.63564	0.02516	0.47445	258s

Figure 111 Multilabel model alternatives

We will take a glance at the overall comparison between binary and multilabel systems. The time execution is higher in the multilabel models. This is due to the dataset being larger and denser, since the model must track six different targets instead of one in the case of binary models. The measures are consistently lower. Nevertheless, we have to be careful about making bold statements about these models being considerably worse than the binary models. Bear in mind that some metrics in multilabel classification may be misleading. This issue is commented in section 7.3.5.7.

The first approach was to explore the limits of Scikit Learn regarding multilabel classification. The library called “skmultilearn” was found during implementation. This library offers the possibility of transforming multilabel problems into binary problems, which are suitable to Scikit Learn. Binary relevance, classifier chain and label powerset are concepts explained in the theoretical aspect sections 2.3.1.6.1, 2.3.1.6.2, and 2.3.1.6.3. These methods perform similarly, but we can observe that the label powerset method lasts much longer, given that it computes every possible combination of targets.

Then, the One vs Rest approach was attempted for the multilabel model. The value for these metrics is the average of all the values of the corresponding metric for each label. Figure 112, Figure 113, Figure 114, and Figure 115 show the specific labels’ metric values of each OvR model.

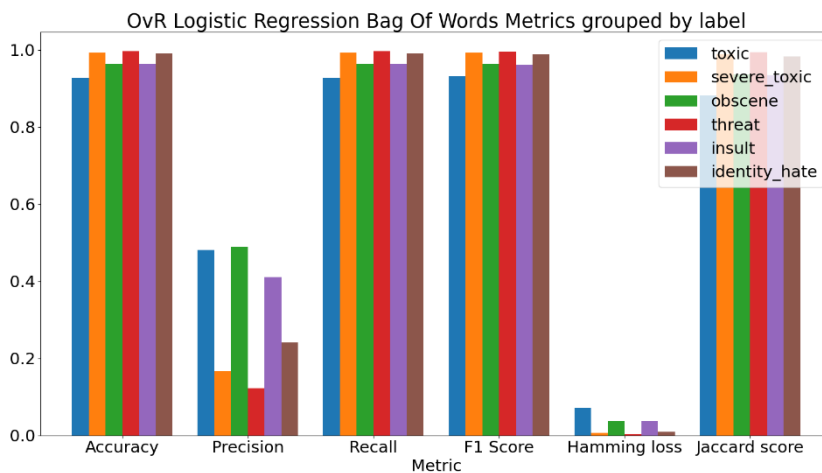


Figure 112 Labels' metric values for model 4 with Bag of words

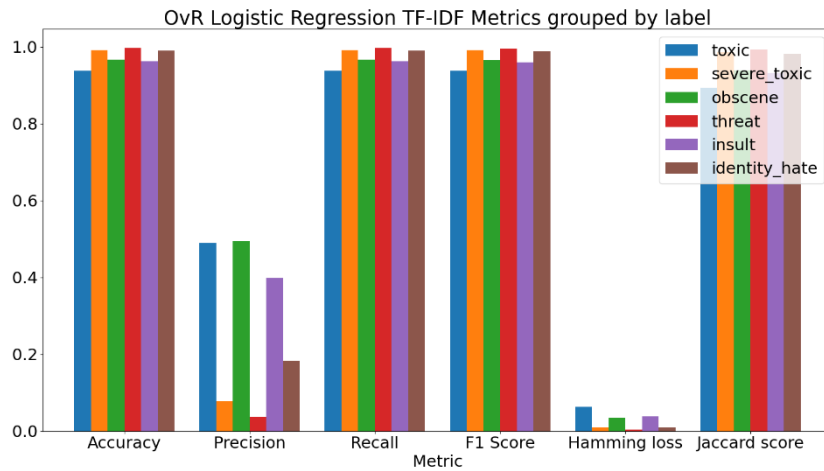


Figure 113 Labels' metric values for model 4 with TF-IDF

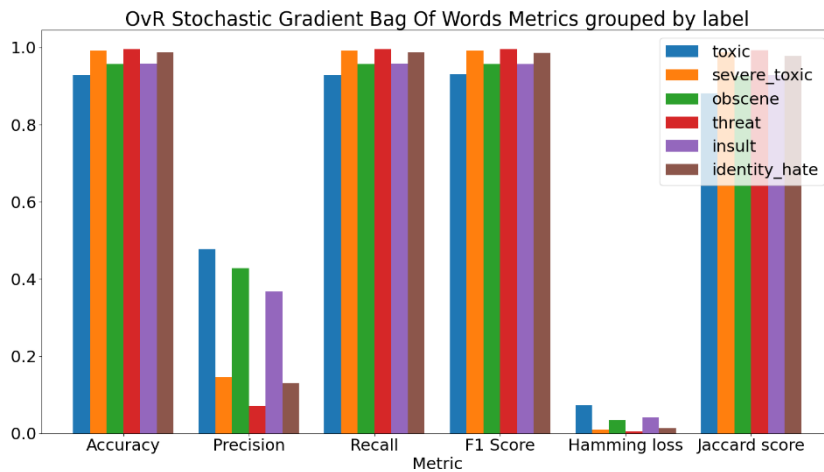


Figure 114 Labels' metric values for model 5 with Bag of words

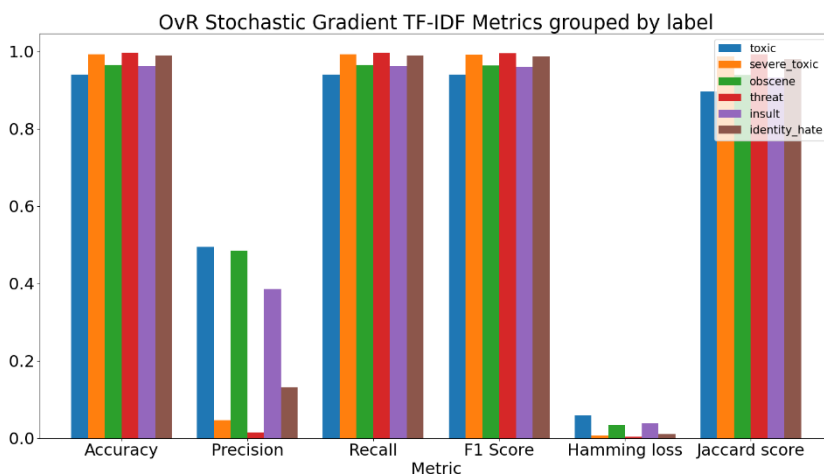


Figure 115 Labels' metric values for model 5 with TF-IDF

We can see that these models performed considerably better than the rest of models. However, their precision values are the lowest (outside the neural networks ones). The value of the

precision metric is calculated as $\frac{TP}{TP+FP}$, so the main concern of this metric is to penalize wrong predictions. Thus, these models tend to predict most of the messages as appropriate. This is probably due to the dataset imbalance, which is commented in section 7.3.5.2.

As one of the main objectives was to create a multilabel mode, the first thought was to implement an artificial neural network. This was done through PyTorch. Only three neural networks were able to complete their training. Each of them has been trained over four 20-hour epochs (iterations). The first two neural networks have not accomplished any metric value higher than 0. This is due to several factors that are discussed in section 7.3.5.4. For the final neural network, a model tried found in [59] was implemented. This is the only deep learning model to perform greatly on the dataset. However, it lacks great values on recall and F1 score metrics.

Finally, both best performing binary models were put to test inside a Multitoutput wrapper that Scikit Learn offers. For the data transformation, a GridSearchCV [94] was performed. This algorithm looks for the best performing parameters inside the data transformation method, in this case TF-IDF, for a given metric. These last models are the ones who perform the most consistently throughout all metrics. These metrics are similar to the first three models', but they perform the training ten times faster than the binary relevance and classifier chain and twenty times faster than the label powerset.

The final model was **10**, given that it performed as the most consistent model. Deciding between **9** and **10** was harder, because the statistical differences are small. However, the SGD classifier provides a `partial_fit` method that allows to retrain the model without losing previous coefficients [26]. The logistic regression does not offer this possibility, so every prediction correction or retraining would have to be done appending it to the original dataset and passing it to the classifier, so response time is the key factor.

Chapter 9. Testing development

The results of the different test are shown in this chapter.

9.1 Unit tests

These are the results of all the test cases listed in section 6.6.1.

<i>Use case 1: Change classification method</i>		
Test 1.1	Expected Result	Passed
The user changes the method to any of the two possible methods	The controller stores the new in the <code>classifier</code> attribute	True
Test 1.2	Expected Result	Passed
The user changes the method to a not contemplated method	The controller itself validates that the value is correct	True

Figure 116 Change classification method unit test result

<i>Use case 2: Detect inappropriate messages</i>		
Test 2.1	Expected Result	Passed
The user tries to classify a blank message with no file uploaded	The controller checks for each mechanism the emptiness of the message. If it is empty an exception is thrown	True
Test 2.2	Expected Result	Passed
The user tries to classify a message that exceeds maximum message length (500 characters)	The controller validates that the message is at most 500-character long. If it is longer, an exception is thrown	True
Test 2.3	Expected Result	Passed
The user tries to upload a file with wrong extension	The controller validates that the extension is correct. If it is not, an exception is thrown	True
Test 2.4	Expected Result	Passed
The user tries to upload with unexpected data	The controller validates the number of columns of the file, which should contain only one The controller validates for every row of the file that the message is a valid character string, that it is not blank and that it does not exceed maximum possible length	True

	For any reason above not being fulfilled, an exception is thrown for each message	
Test 2.5	Expected Result	Passed
The user types a message with no file uploaded and classifies the message	The controller computes the prediction The user interface shows the prediction	True
Test 2.6	Expected Result	Passed
The user uploads a correct file with valid data and with no typed message	The controller predicts for each message a prediction as the one described before	True

Figure 117 Detect inappropriate messages unit test result

<u>Use case 3: Save results to file</u>		
Test 3.1	Expected Result	Passed
The user tries to save results to file with no classification performed	The controller raises an exception	True
Test 3.2	Expected Result	Passed
The user saves results of a performed classification to a file	The controller creates a file inside the output folder, and inserts the last prediction, dumping the message and the obtained prediction For several predictions, each line of the output file will contain the message and the obtained prediction	True
Test 3.3	Expected Result	
The user tries to save results to .txt file with no classification performed	The controller raises an exception	True
Test 3.4	Expected Result	
The user saves results to .txt of a performed classification to a file	The controller creates a file inside the destination folder, and inserts the last prediction, dumping the message and the obtained prediction For several predictions, each line of the output file will contain the message and the obtained prediction	True

Figure 118 Save results to file unit test result

Use case 4: Correct predictions

Test 4.1	Expected Result	Passed
The administrator tries to correct a prediction with no classification performed	The controller checks if the <code>last_predictions</code> attribute is empty	True
Test 4.2	Expected Result	Passed
The administrator introduces a correction with wrong values or number of values	The controller raises an exception	True
Test 4.3	Expected Result	Passed
The administrator introduces a correct prediction	There are no errors in the return value	True
Test 4.4	Expected Result	Passed
A non-administrator user tries to correct the prediction	The controller checks if the user is logged in	True

Figure 119 Correct predictions unit test result

<u>Use case 5: Log in as administrator</u>		
Test 5.1	Expected Result	Passed
The administrator tries to access the Log in as administrator option	The controller checks that the user is not already logged in	True
Test 5.2	Expected Result	Passed
The user tries to introduce blank username or password	The controller validates if any of the data is blank. If it is, the controller raises an exception	True
Test 5.3	Expected Result	Passed
The user tries to introduce too long username (20 characters) or password (30 characters)	The controller checks that the username and password do not exceed the possible maximum length (20 and 30 characters respectively). If any does, the controller throws an exception	True
Test 5.4	Expected Result	Passed
The user tries to introduce non-alphanumeric characters for the username	The controller checks that the username is only alphanumeric. If it is not, the controller throws an exception	True
Test 5.5	Expected Result	Passed
The user introduces valid username and password but there	The authentication module returns a negative result (<code>false</code>) because the	True

is not match in the database	hashed password and the database password do not match	
Test 5.6	Expected Result	
The user introduces valid and correct username and password	The authentication module returns a positive result (<code>true</code>) The attribute <code>authenticated</code> in the controller is set to <code>true</code>	

Figure 120 Log in as administrator unit test result

Use case 6: Train models		
Test 6.1	Expected Result	Passed
The non-administrator user tries to train a model	The controller checks if the user is logged in	True
Test 6.2	Expected Result	Passed
The administrator tries to upload an invalid file	The controller checks the extension of the file. If it is incorrect, the controller raises an exception	True
Test 6.3	Expected Result	Passed
The administrator uploads a file with blank data or wrong prediction values	The classifier checks the validity of the data. If it is not valid, an exception is thrown for each of the rows	True
Test 6.4	Expected Result	Passed
The administrator uploads a file with wrong number of pieces of data	The classifier checks the number of columns of the file. If it is incorrect, an exception is thrown	True
Test 6.5	Expected Result	Passed
The administrator uploads a valid file with valid data and trains the model	There are no errors in the return value	True

Figure 121 Train models unit test result

9.2 Usability tests

Now, the results of the surveys completed by the testing user will be shown. Bold options represent the user answer. The tests have been complimented by three people with different computer experience.

9.2.1 User profile ranking

Each of the users has completed this survey.

Name: Jorge Antonio
How often do you use a computer?

<ol style="list-style-type: none"> 1. Everyday 2. Several times a week 3. Occasionally 4. Hardly ever 5. I have never used a computer
What is your main activity using a computer?
<ol style="list-style-type: none"> 1. It is part of my job or occupation 2. Mainly for free time 3. Using office software 4. Reading news and/or emails 5. I use it for nearly everything
Have you ever used a similar software?
<ol style="list-style-type: none"> 1. Yes, I have 2. No, although I have used software that perform similar tasks 3. No
What do you look forward the most in a program?
<ol style="list-style-type: none"> 1. To be easy to use 2. To perform a lot of tasks 3. To be fast 4. To have a nice interface 5. To be transparent on the operations it performs

Figure 122 User profile ranking survey completed by Jorge Antonio

Name: Marta
How often do you use a computer?
<ol style="list-style-type: none"> 1. Everyday 2. Several times a week 3. Occasionally 4. Hardly ever 5. I have never used a computer
What is your main activity using a computer?
<ol style="list-style-type: none"> 1. It is part of my job or occupation 2. Mainly for free time 3. Using office software 4. Reading news and/or emails 5. I use it for nearly everything
Have you ever used a similar software?
<ol style="list-style-type: none"> 1. Yes, I have 2. No, although I have used software that perform similar tasks

3. No
What do you look forward the most in a program?
<ol style="list-style-type: none"> 1. To be easy to use 2. To perform a lot of tasks 3. To be fast 4. To have a nice interface 5. To be transparent on the operations it performs

Figure 123 User profile ranking survey completed by Marta

Name: Jose Ignacio
How often do you use a computer?
<ol style="list-style-type: none"> 1. Everyday 2. Several times a week 3. Occasionally 4. Hardly ever 5. I have never used a computer
What is your main activity using a computer?
<ol style="list-style-type: none"> 1. It is part of my job or occupation 2. Mainly for free time 3. Using office software 4. Reading news and/or emails 5. I use it for nearly everything
Have you ever used a similar software?
<ol style="list-style-type: none"> 1. Yes, I have 2. No, although I have used software that perform similar tasks 3. No
What do you look forward the most in a program?
<ol style="list-style-type: none"> 1. To be easy to use 2. To perform a lot of tasks 3. To be fast 4. To have a nice interface 5. To be transparent on the operations it performs

Figure 124 User profile ranking survey completed by Jose Ignacio

9.2.2 Guided activities

Every user has complimented this questionnaire.

Name: Jorge Antonio
Writing a message and predicting it

<p><i>Things I liked:</i> The option to itemize a prediction into several categories</p> <p><i>Things I would like to be improved:</i> Model names in the user interface could be clearer</p>
<p>Uploading a file with messages and predicting them</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> I don't like using double quotes to write the messages</p>
<p>Obtaining a file with the computed predictions</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> -</p>
<p>Correcting the prediction of a computed classification (Administrators only)</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> Progress bar so I know I have to wait</p>
<p>Uploading a file to train a model (Administrators only)</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> Help button to know the file format</p>

Figure 125 Guided activities survey completed by Jorge Antonio

<p>Name: Marta</p>
<p>Writing a message and predicting it</p> <p><i>Things I liked:</i> Easy to know how to use it</p> <p><i>Things I would like to be improved:</i> To be able to edit the message after classifying it</p>
<p>Uploading a file with messages and predicting them</p> <p><i>Things I liked:</i> Also, easy to use</p> <p><i>Things I would like to be improved:</i> It would be nice if the program accepted other more human-friendly extensions like .txt</p>
<p>Obtaining a file with the computed predictions</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> The task would be better if it let you select the path</p>
<p>Correcting the prediction of a computed classification (Administrators only)</p> <p><i>Things I liked:</i> -</p> <p><i>Things I would like to be improved:</i> The Correct prediction text should be bigger</p>
<p>Uploading a file to train a model (Administrators only)</p> <p><i>Things I liked:</i></p>

- Things I would like to be improved: As uploading a message, it would be better if it let you submit other extensions

Figure 126 Guided activities survey completed by Marta

Name: Jose Ignacio
Writing a message and predicting it
Things I liked: Easy to use Things I would like to be improved: -
Uploading a file with messages and predicting them
Things I liked: - Things I would like to be improved: The message about file submission should be clearer
Obtaining a file with the computed predictions
Things I liked: - Things I would like to be improved: -
Correcting the prediction of a computed classification (Administrators only)
Things I liked: The confirmation window where I can see the overall operation Things I would like to be improved: -
Uploading a file to train a model (Administrators only)
Things I liked: The confirmation window where I can see the overall operation Things I would like to be improved: I would want to be provided an example inside the program on how each row should be

Figure 127 Guided activities survey completed by Jose Ignacio

9.2.3 Quick questions about the application

All users have done the following survey.

Name: Jorge Antonio				
Ease of use	Always	Most of the time	Occasionally	Never
Do you know where you are inside the application?		X		
Is there any help when using the application if any doubts arise?			X	
Do you consider the application to be easy to use?		X		
Functionality	Always	Most of the time	Occasionally	Never
Does the prediction suit the messages introduced?			X	

Does the model reflect a better prediction after correcting or training the model?		X		
Is the files' data format intuitive and suitable?		X		
Does every task work as expected?	X			
If any task fails, do you find the error messages descriptive enough?		X		
Is the response time of the application adequate?		X		
Interface quality				
Graphic aspects	Very adequate	Adequate	Little adequate	Not adequate
The font size and type is		X		
Used colours are			X	
Interface design		Yes	No	Sometimes
Is the interface easy to use?		X		
Is the windows design clear?		X		
Do you think that the application is well-structured?		X		
Are messages describing thoroughly the specific situation?		X		
Observations				
The user interface and colours are a little bit cold and could be improved				

Figure 128 Quick question about the application survey completed by Jorge Antonio

Name: Marta				
Ease of use	Always	Most of the time	Occasionally	Never
Do you know where you are inside the application?		X		
Is there any help when using the application if any doubts arise?			X	
Do you consider the application to be easy to use?	X			
Functionality	Always	Most of the time	Occasionally	Never
Does the prediction suit the messages introduced?		X		
Does the model reflect a better prediction after correcting or training the model?	X			
Is the files' data format intuitive and suitable?		X		
Does every task work as expected?	X			
If any task fails, do you find the error messages descriptive enough?		X		

<i>Is the response time of the application adequate?</i>		X		
Interface quality				
Graphic aspects	Very adequate	Adequate	Little adequate	Not adequate
<i>The font size and type is</i>			X	
<i>Used colours are</i>		X		
Interface design		Yes	No	Sometimes
<i>Is the interface easy to use?</i>		X		
<i>Is the windows design clear?</i>		X		
<i>Do you think that the application is well-structured?</i>		X		
<i>Are messages describing thoroughly the specific situation?</i>		X		
Observations				
I have understood how the application works with no previous knowledge on artificial intelligence and I'd probably use it in some of my projects.				

Figure 129 Quick question about the application survey completed by Marta

Name: Jose Ignacio				
Ease of use	Always	Most of the time	Occasionally	Never
<i>Do you know where you are inside the application?</i>		X		
<i>Is there any help when using the application if any doubts arise?</i>			x	
<i>Do you consider the application to be easy to use?</i>	X			
Functionality	Always	Most of the time	Occasionally	Never
<i>Does the prediction suit the messages introduced?</i>		X		
<i>Does the model reflect a better prediction after correcting or training the model?</i>		X		
<i>Is the files' data format intuitive and suitable?</i>			X	
<i>Does every task work as expected?</i>	X			
<i>If any task fails, do you find the error messages descriptive enough?</i>	X			
<i>Is the response time of the application adequate?</i>			X	
Interface quality				
Graphic aspects	Very adequate	Adequate	Little adequate	Not adequate
<i>The font size and type is</i>		X		
<i>Used colours are</i>		X		
Interface design		Yes	No	Sometimes
<i>Is the interface easy to use?</i>		X		
<i>Is the windows design clear?</i>		X		

Do you think that the application is well-structured?	X		
Are messages describing thoroughly the specific situation?	X		
Observations			
-			

Figure 130 Quick question about the application survey completed by Jose Ignacio

9.2.4 Tester survey

The developer responsible of creating and distributing the test has completed the next summary of the testing phase.

Observed aspect	Notes	Possible solutions
<i>The user handles the task in a fast way</i>	All users have asserted that the application performs tasks quickly	
<i>Minor errors</i>	In 9.2.2, an error that allowed pressing the Classify option several times for the same message was found	The Classify option is now only available once after writing or uploading a file
<i>Major errors</i>	N/D	
<i>The predictions are fitting to the true ones</i>	Users that performed predictions in 9.2.2 had shown that the predictions are acceptable, but could be better	A deep learning model could generate better results, but the problems listed in 7.3.5 have made a deep learning model unreachable. So, the current models are kept
<i>Files' data format is intuitive to the user</i>	Users agree that data's format is proper. However, some complains suggested to also accept .txt files	As the most agreed format for machine learning datasets is .csv files containing rows of data, this has not been changed
<i>Output files show content clearly and plainly</i>	Users demanded more extension like .txt	The application now offers a .txt saving option
<i>User interface is suitable</i>	Users have accepted the user interface style. However, there were some complains about font size and style	The interface has been altered to satisfy these complains
	The window should be possible to be maximized	This is a constraint given by the user interface library, so it has not been changed
	When doing the training, an example should be shown in the interface	A message is shown now explaining an example of how a file row should be written
	Some users have demanded an option to refresh the prediction without writing or submitting the message again	A functionality has been added in the logic that allows to make the last predictions performed

	Users had asked for any kind of help within the application for format of the message and training files	Two options have been added to the final design of the user interface which display the format of these files
--	--	---

Table 1 Completed tester survey

9.3 Accessibility tests

Given that the developed system is a desktop application and in order to prove the accessibility of the application, a new customized checklist acquired from alterations of the WCAG 1.0 standards is complimented [39]. For the contrast ratio, we can take a look at [95].

Checklist points	Achieved	Not achieved
Sensory characteristics (A) [96]		
Do not identify content based on its colour, size, shape, position, sound, or other sensory characteristics	X	
Do no convey information solely through icons or symbols	X	
Use of colour (A) [97]		
Required fields and fields with errors must include some non-color way to identify them	X	
Contrast (Minimum (AA) [98]		
Text (including images of text) have a contrast ratio of at least 4.5:1. For text and images of that is at least 24px and normal weight or 19px and bold, use a contrast ratio that is at least 3:1.	X	
Text spacing (AA) [99]		
Avoid using pixels for defining the height and spacing of text boxes	X	
Keyboard (A) [100]		
All functionalities should be available to a keyboard without requiring specific timing of keystrokes, unless the functionality cannot be provided by a keyboard alone	X	
No keyboard trap (A) [101]		
Ensure keyboard focus is never trapped on an element without an obvious way to move focus out of the element. Make sure the user can move focus to and from all focusable elements using a keyboard only		X
Timing adjustable (A) [102]		
Do not require time limits to complete tasks unless absolutely necessary. If a time limit is necessary, the time limit should be at least 20 hours, or it can be extended, adjusted, or disabled.	X	

Pause, Stop, Hide (A) [103]		
Items on the page should not automatically move, blink, scroll, or update, including carousels. If content does automatically move, blink, scroll, or update, provide a way to pause, stop, or hide the moving, blinking, scrolling, or updating.	X	
Focus Visible (A) [104]		
Provide keyboard focus styles that are highly visible, and make sure that a visible element has focus at all times when using a keyboard. Do not rely on browser default focus styles.	X	
Pointer Cancellation (A) [105]		
Avoid triggering functionality on down-events, such as onmousedown. Use events such as onclick instead.	X	
Label in Name (A) [106]		
The accessible name for a UI element must contain any visual label for the element. Accessible names for UI elements should match visual labels as closely as possible.	X	
On Input (A) [107]		
When a user inputs information or interacts with a control, the window should not cause a change in page content, spawn a new browser window, submit a form, cause further change in focus, or cause any other change that disorients the user. If an input causes such a change, the user must be informed ahead of time.	X	
Error Identification (AA) [108]		
Make errors easy to discover, identify, and correct	X	
Labels or Instructions (A) [109]		
Use semantic, descriptive labels for inputs. Visually position labels in a consistent way that makes associating labels with form controls easy	X	
Provide text instructions at the beginning of a form or set of fields that describes the necessary input.	X	
Error Suggestion (AA) [110]		
If an input error is detected and if suggestions for correction are known, provide suggestions for fixing the submission.	X	

Figure 131 Customized accessibility checklist

9.4 Performance tests

These are the results of the fulfilled performance tests listed in section 6.6.5.

Test	Workload	Result
1.1 Initial training of binary model	Binary dataset [40]	RAM Usage: 274.3 MiB Time elapsed: 66.96s
1.2 Initial training of multilabel model	Multilabel dataset [41]	RAM Usage: 503.1 MiB Time elapsed: 269.29s
1.3 Predict messages (binary)	10000	RAM Usage: 239.32 MiB Time elapsed: 20.07s
1.4 Predict messages (multilabel)	10000	RAM Usage: 219.24 MiB Time elapsed: 81.2s
1.5 Predict messages from file (binary)	10000	RAM Usage: 239.2 MiB Time elapsed: 16.45s
1.6 Predict messages from file (multilabel)	10000	RAM Usage: 220.23 MiB Time elapsed: 21.36s
1.7 Saving predictions to file .csv	10000	RAM Usage: 238.99 MiB Time elapsed: 0.2s
1.8 Saving predictions to file .txt	10000	RAM Usage: 239.02 MiB Time elapsed: 0.22s

Figure 132 Performance tests results

Chapter 10. System manuals

This chapter presents the different manuals available about the system, for non-administrator users, administrators, and programmers.

10.1 Installation manuals

The system is attached as a compressed file inside the delivered content. First, we should unzip this file any directory. It will create the folder `hate-speech-detection` which contains all the needed files to run the system.

Before running anything, we need the 3.10.2 Python version in order to execute the program. It can be downloaded in [42]. It is important that we select the `Add Python 3.10 to PATH` so we can execute the command outside the installation directory.

Then, we can open a command-line prompt inside the `hate-speech-detection` folder, by writing `cmd` in the file explorer and press Enter.

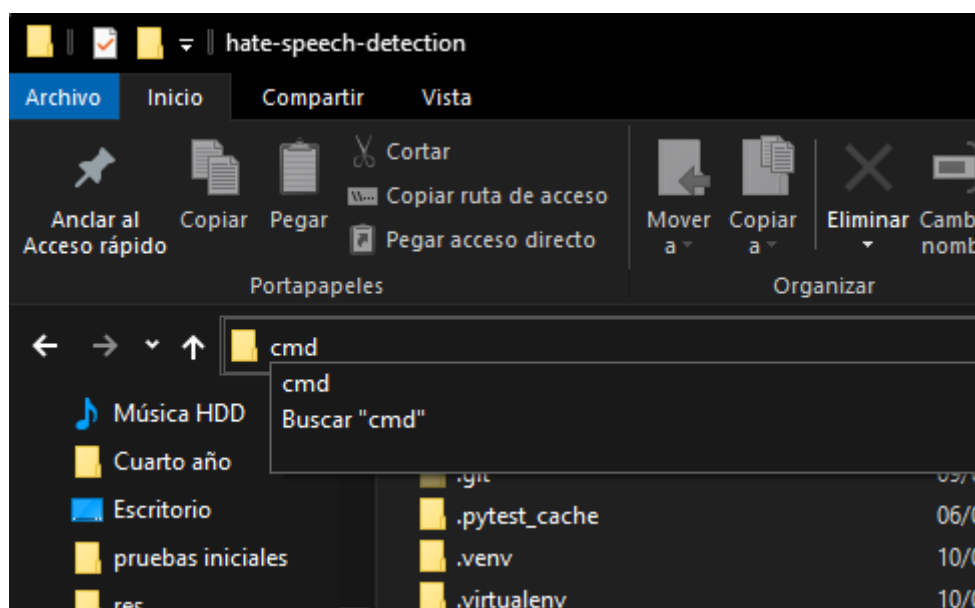


Figure 133 Installation manual Open CMD

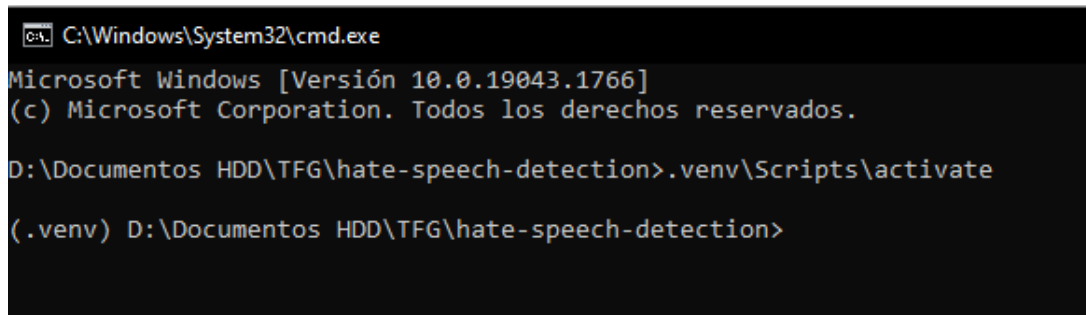
In order to encapsulate all the following installations, we create a virtual environment to contain the modules. If Python has been well configured, the command to run would be:

```
python -m venv .venv
```

This creates a folder named `.venv` inside the project directory that contains some Python related files. We have to execute a file inside this new folder with the following command:

```
.venv\Scripts\activate
```

Now we should have a `(.venv)` before the current directory.



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1766]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\Documentos HDD\TFG\hate-speech-detection>.venv\Scripts\activate

(.venv) D:\Documentos HDD\TFG\hate-speech-detection>

```

Figure 134 Installation manual Virtual environment activation

Now we will install all the needed modules for the program to be run.. These modules are listed in the `requirements.txt` file provided in the attached content ready to be passed as an argument for a suitable command. The command to be run is:

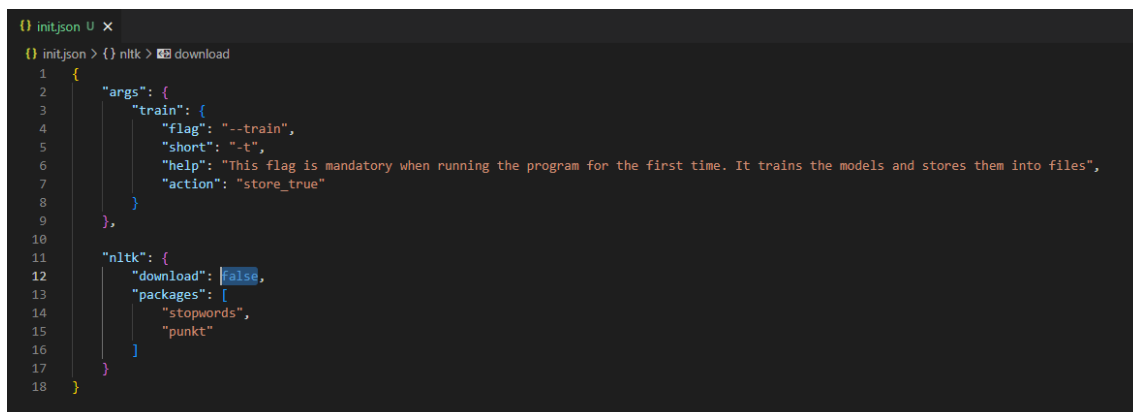
```
pip install -r requirements.txt
```

10.2 Execution manuals

After all the installation steps in 10.1, we can run the program. The executing file is `main.py`, which is inside the `src/` folder. If we are in the project root folder the command to execute the program is

```
python .\src\main.py
```

This should make a window appear after a few seconds and we can now interact with the application. For the subsequent executions of the system, we should modify the `init.json` file and set the `download` option inside the `nlTK` option to `false`. This step is not mandatory, but it is recommended for efficiency reasons.



```

init.json U x
init.json > {} nlTK > download
1
2 {
3   "args": {
4     "train": {
5       "flag": "--train",
6       "short": "-t",
7       "help": "This flag is mandatory when running the program for the first time. It trains the models and stores them into files",
8       "action": "store_true"
9     }
10  },
11  "nlTK": {
12    "download": false,
13    "packages": [
14      "stopwords",
15      "punkt"
16    ]
17  }
18 }

```

Figure 135 Execution manual init.json

10.3 User manual

This manual will in detail describe the steps to execute the possible tasks that the program offers. These tasks are the ones listed:

- Choose a classification method (Binary method or itemized method).
- Predict a message.
- Predict messages in a file.
- Save results to the file.
- Log in as administrator (Non-administrator users only).
- Correct predictions of a performed classification (Administrators only).
- Train any of the models with new data (Administrators only).

All the tasks but the last two are doable for every user and will be described in 10.3.1. Then, the Administrator manual will be presented explaining the two last operations.

10.3.1 Non-administrator user manual

In this manual, the tasks of changing the classification method, predicting messages, and saving the predictions to a file are detailed.

10.3.1.1 Start of the application

In case we have run the program for the first time in our local machine with the `-t` option, it will take around 7-10 minutes for the models to be created and stored. After that, the interface presented will look like this:

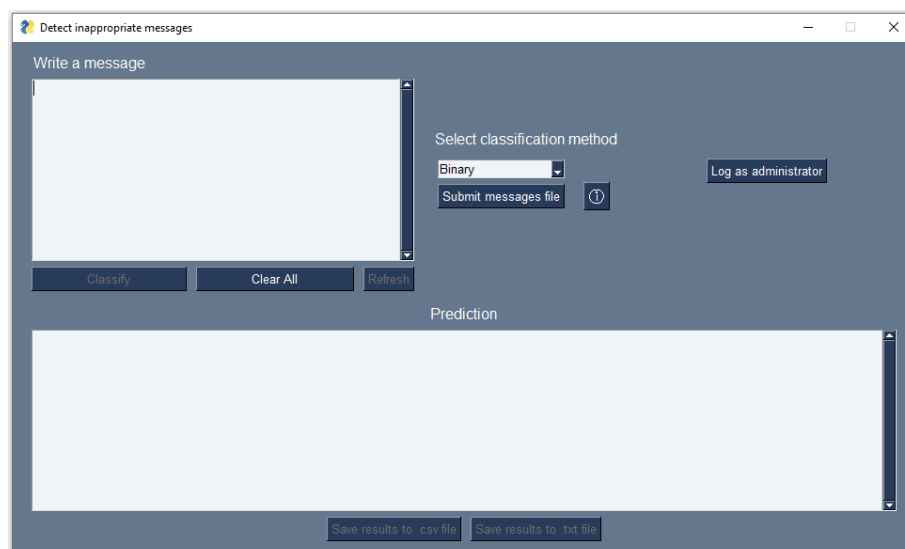


Figure 136 Start of application

The first text area allows the user to type in any message. There are also options to change the method for classifying the messages, a button to submit a file with messages, a button to start over another classification and an option to log as an administrator. The text area below will show the prediction results.

10.3.1.2 Predicting a message

To perform a classification of a single message, the user can type in any text in the first text area.

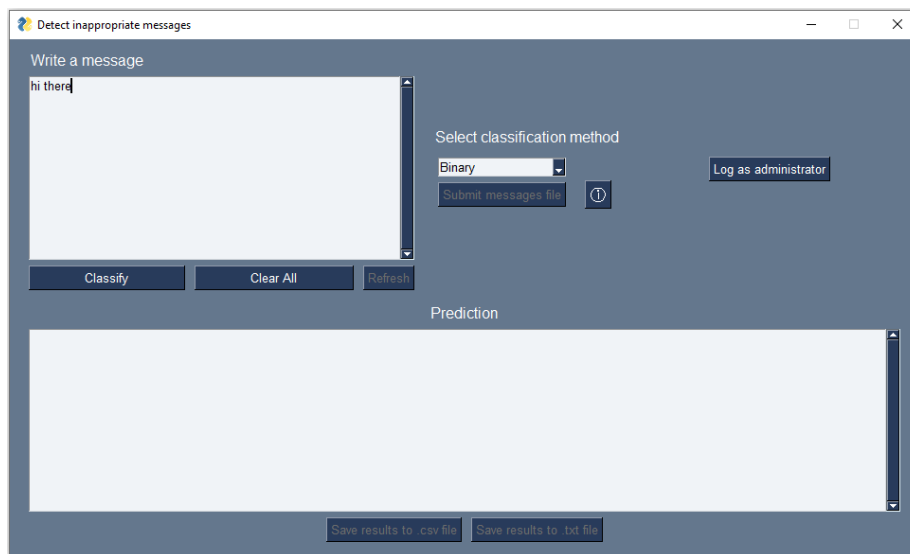


Figure 137 Step 1 to predict a message

After writing the message, press the Classify button. This will show in the second text area the number of the message (the user may see that his message has a number attached to the left) and the message corresponding prediction. It may show a No prediction text if the message exceeds the number of maximum characters, which is 500.

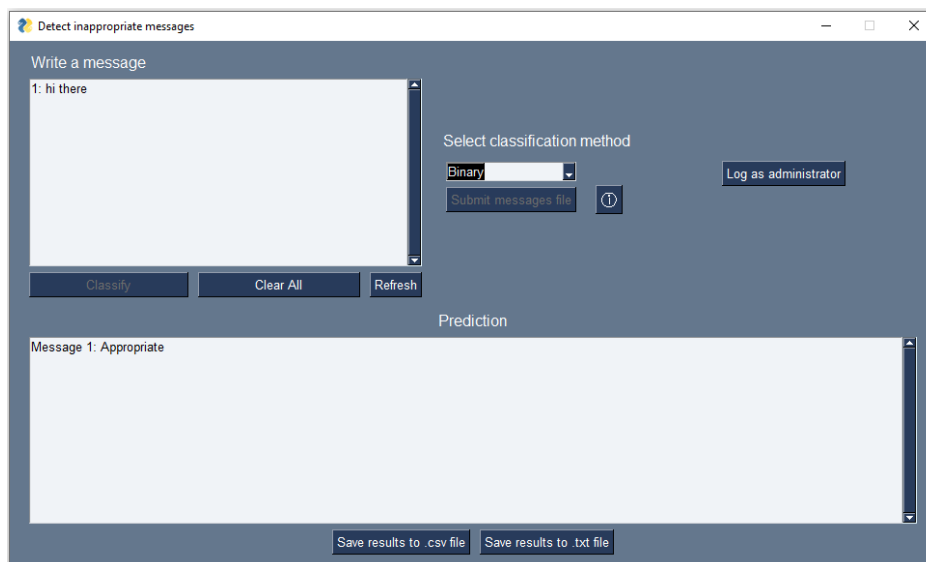


Figure 138 Final step to predict a message

After that, the user may save the result to a file. To perform another classification, press the button Clear All to return to the Start of application. The user can press the refresh button to perform the same operation again.

10.3.1.3 Predict messages in a file

Press the button Submit messages file in order to select a file containing the messages. A file explorer will be open.

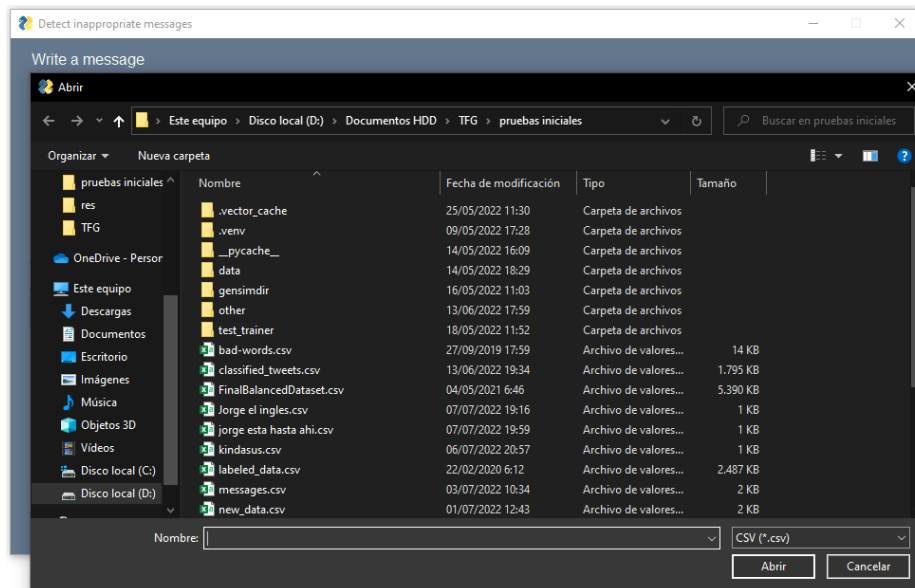


Figure 139 Step 1 to predict messages in a file

The user chooses then the desired file. The file explorer will be closed, and the system will display a message confirming the submission. If the user wants to change the file, it may press the button Clear All and repeat the process.

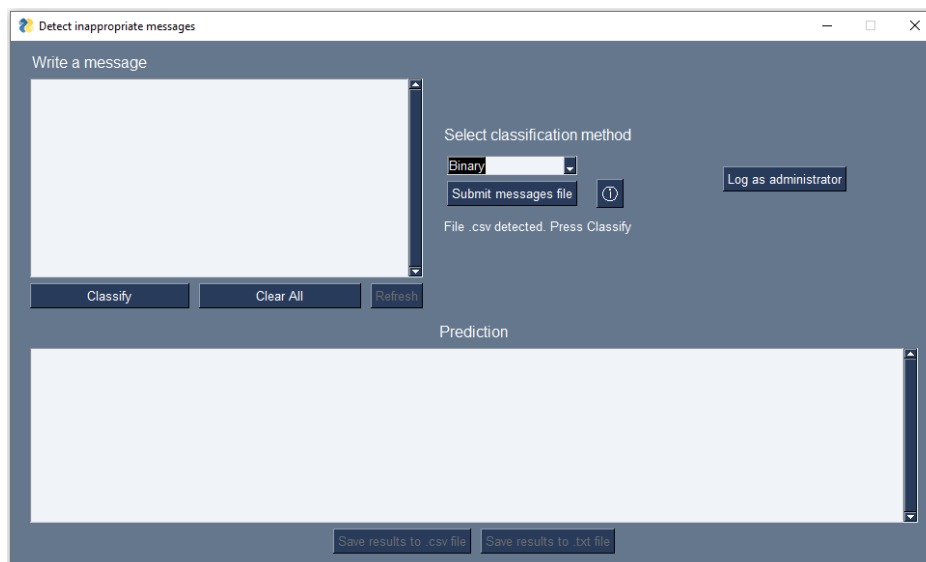


Figure 140 Step 2 to predict messages in a file

Now the user may see the file submission confirmation. The system will enable the Classify option. Press the option and the results will be presented.

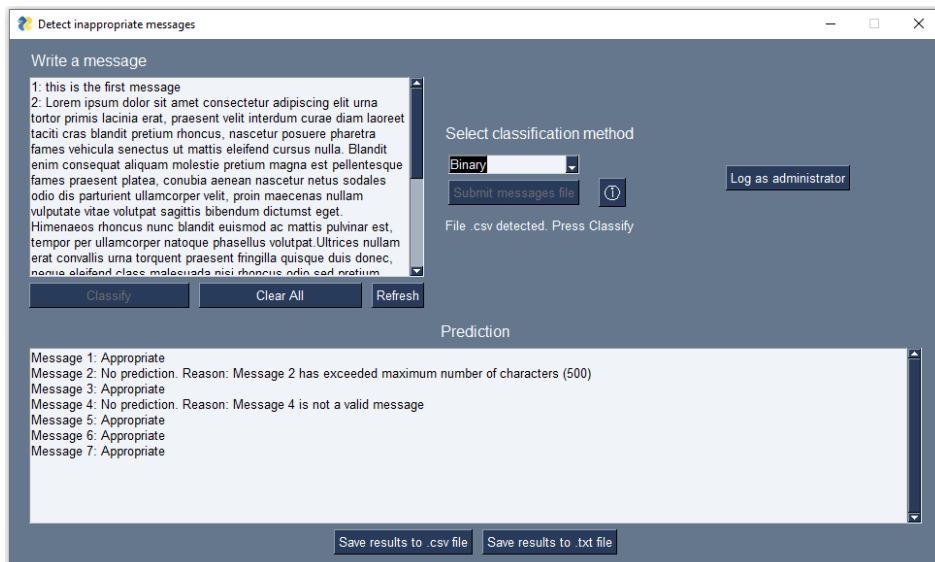


Figure 141 Final step to predict messages in a file

The system will display in the Prediction area each prediction of every message. If the file contained wrong text values, the system would automatically inform the user and predict the rest of valid messages. To perform another classification, press the Clear All option to return to the Start of application.

10.3.1.3.1 File format

The file format is explained in the option with the ⓘ icon.

The system will accept a concrete format for the messages submission file. If it does not fulfil these requirements, the file is discarded.

- The file must have a .csv extension.
- The file must contain a message per row.
- The file must enclose each message between double quotes ("").

10.3.1.4 Change the classification method

The system currently offers two models to classify a message, a binary model classifier and an itemized classifier. The former classifies the message as appropriate or not. The later provides more detail, predicting toxicity, obscenity, threat, insult, and identity hate values. Press the down arrow in the option Select classification method. It will display both options.

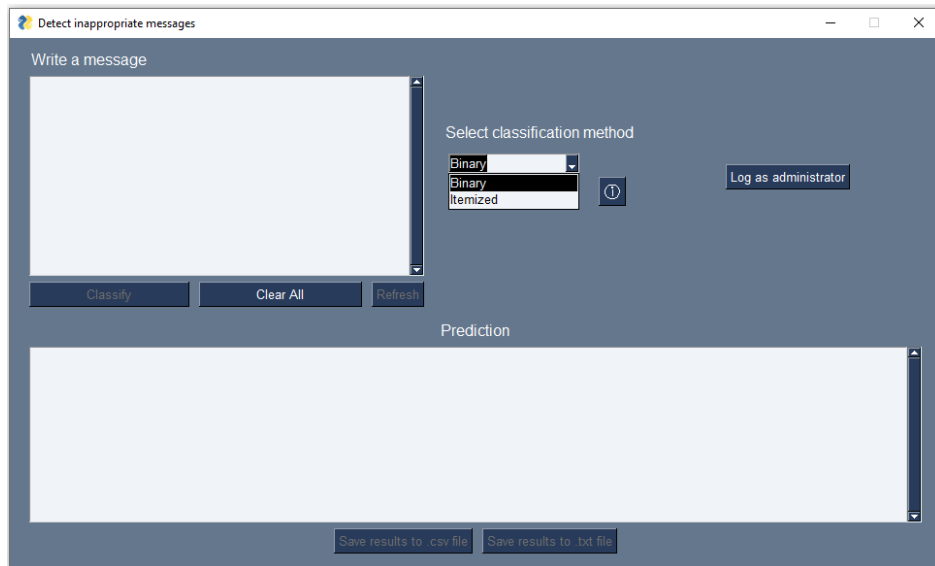


Figure 142 Change the method for classification

This next figure shows an example of a prediction performed with the itemized method.

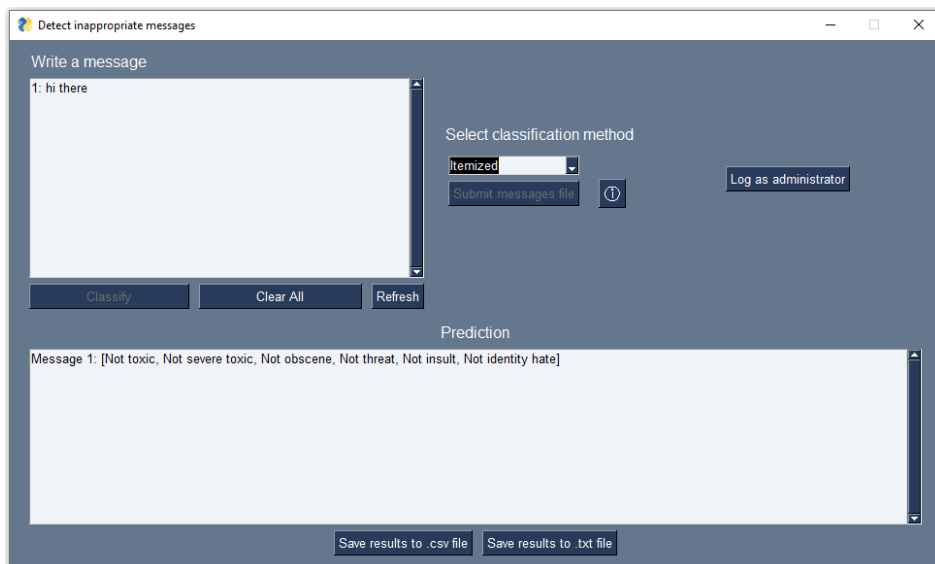


Figure 143 Example of predicting with the itemized method

10.3.1.5 Save predictions to file

The user can save the performed predictions in both .csv and .txt formats. In order to be able to make this operation, the user must have performed a classification as indicated in Figure 138, Figure 141, or Figure 143. The Save results to file options will be available. After pressing any of them, a file explorer will appear to select the destination of the file.

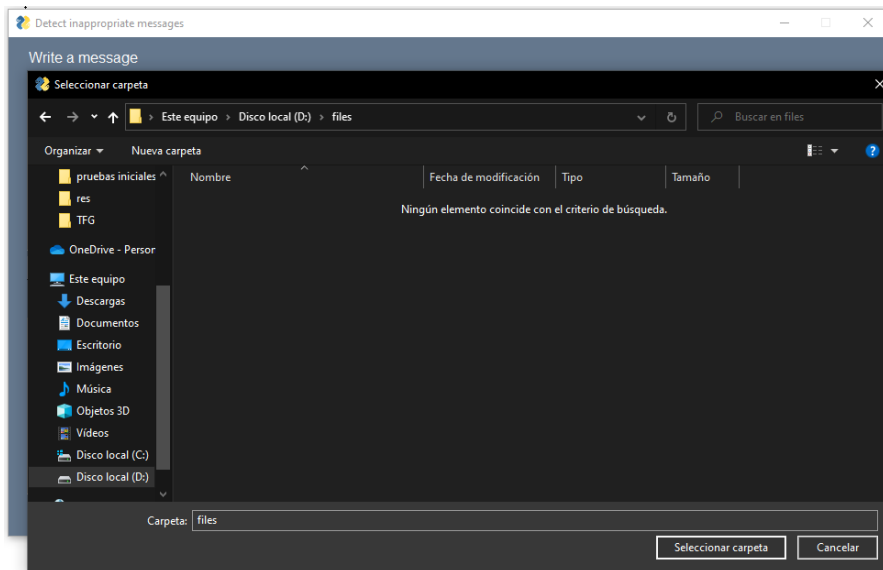


Figure 144 File explorer for saving results to a file

We will be shown a message detailing the name of the file.

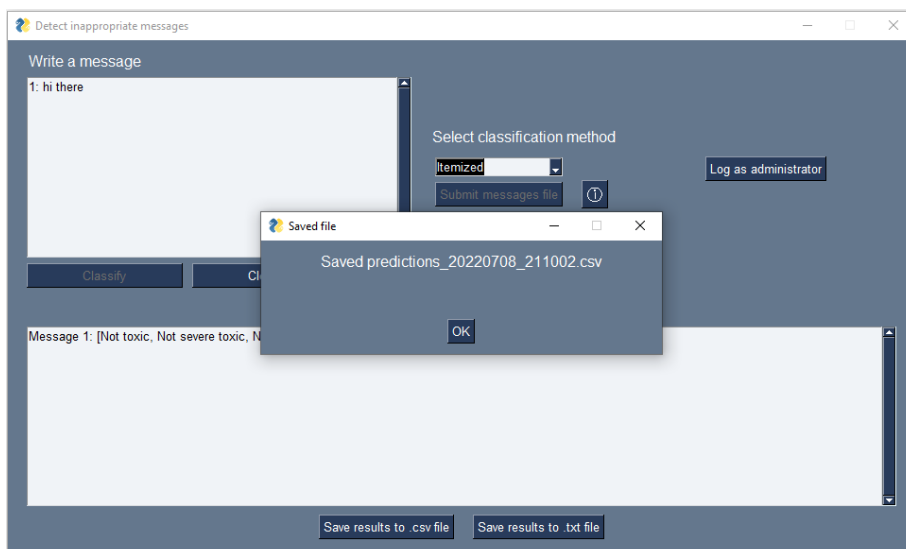


Figure 145 Example of saving prediction to file

If we open this file (named with the current timestamp), we will see the information of the prediction. The .csv file has a format suitable for being input into a dataset. The output shows numbers to represent the prediction. The reason behind this decision is that most of the machine learning models accept number as input to perform classifications. This file would be a perfect input for any machine learning model that manages the same sort of task.

```
predictions_20220708_211002.csv X
predictions_20220708_211002.csv
1 message,toxic,severe_toxic,obscene,threat,insult,identity_hate
2 ""hi there"",0,0,0,0,0,0
3
```

Figure 146 Example of saved prediction .csv file

The .txt file shows a more human-friendly message.

```
predictions_20220708_211022.txt X
predictions_20220708_211022.txt
1 The message "hi there" has been predicted as [Not toxic, Not severe toxic, Not obscene, Not threat, Not insult, Not identity hate]
2
```

Figure 147 Example of saved prediction .txt file

10.3.1.6 Log as administrator

The last available option for a user which is not logged as an administrator is to log in to reach the rest of the possible operations. Press the Log as administrator button to access to the log-in dialog.

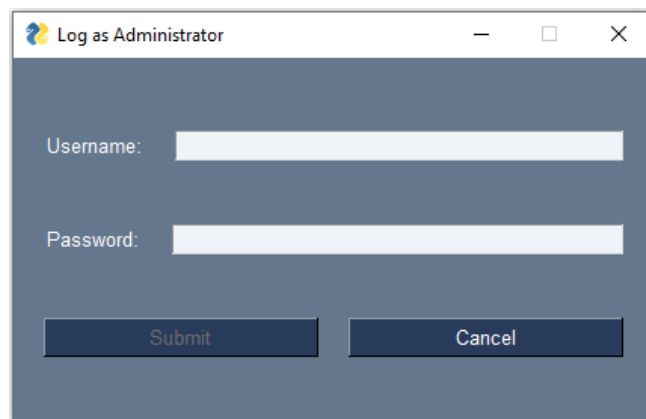


Figure 148 Authentication dialog

The user must introduce a valid username and password to log as an administrator. If any error occurs, the dialog will show the issue to the user. If the authentication is correct, the user will return to the main window as an administrator.

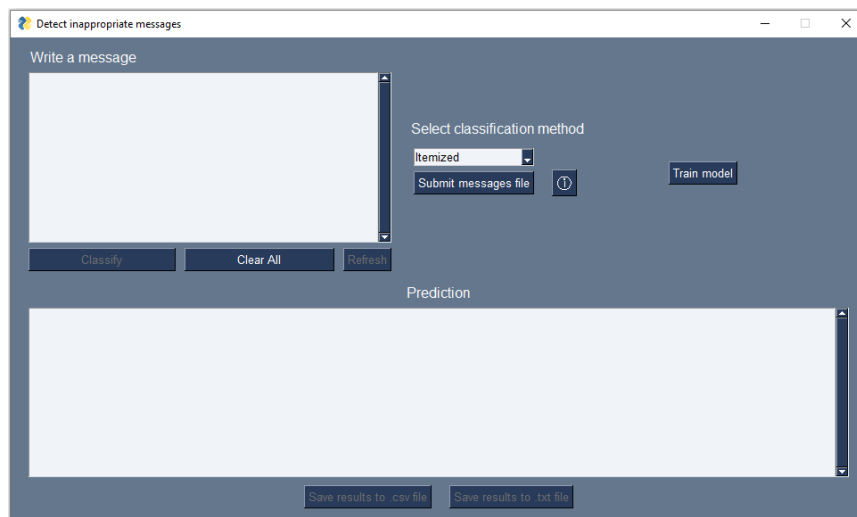


Figure 149 Main window after logging as administrator

The user has the option now of training the models and correcting any prediction performed beforehand.

10.3.2 Administrator manual

After logging in as administrator, the user has the option to alter the models' functionality via training them or correcting predictions of performed classifications.

For logging in as administrator we can use username «admin1» and password «admin1».

10.3.2.1 Correcting a prediction

The administrator must have performed a classification as indicated in Figure 138, Figure 141, or Figure 143. An option to choose the number of a message should appear on the right part of the main window.

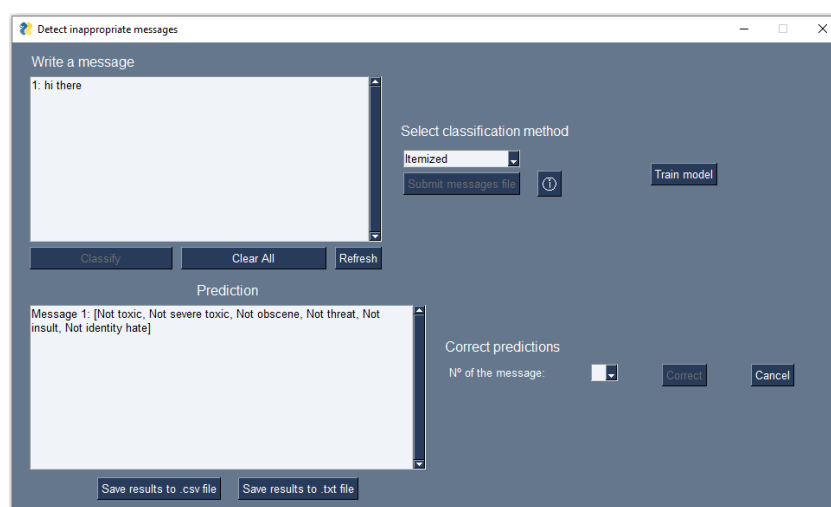


Figure 150 Prediction performed as administrator

If we display the option N° of the message, we will see a list of numbers corresponding to the number of valid predicted messages. The user may select one to alter the prediction.

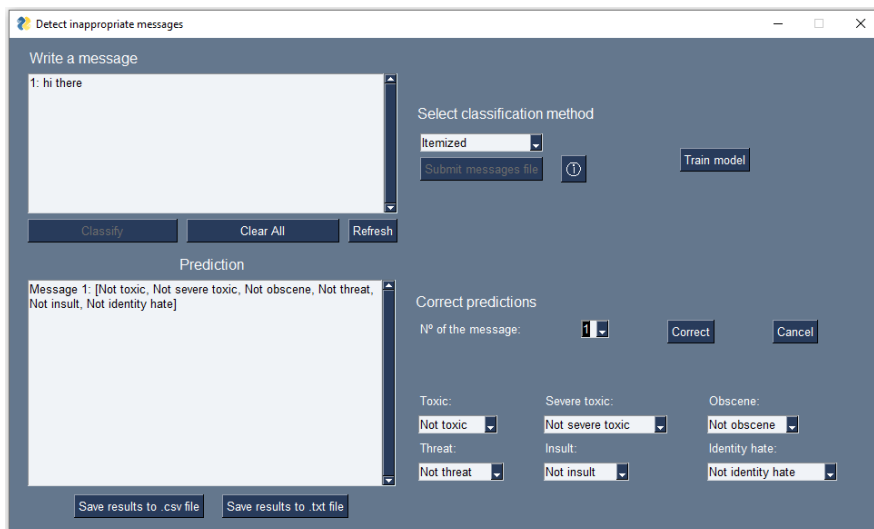


Figure 151 Chosen message to correct prediction

Choose from each option of the prediction the desired value and press Submit. If the operation wants to be cancelled, the user may select the Cancel option.

If the user submits the new prediction values, a new confirmation dialog will be displayed for the process of the operation. It will contain information regarding the correction of the prediction and options for performing or cancelling it. Press Confirm to proceed to the correction.

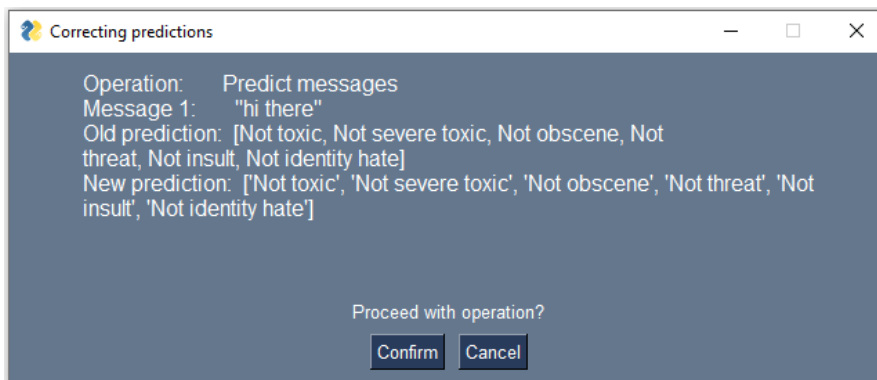


Figure 152 Confirmation window for correction of a prediction

The result is shown after confirming the operation after a few seconds.

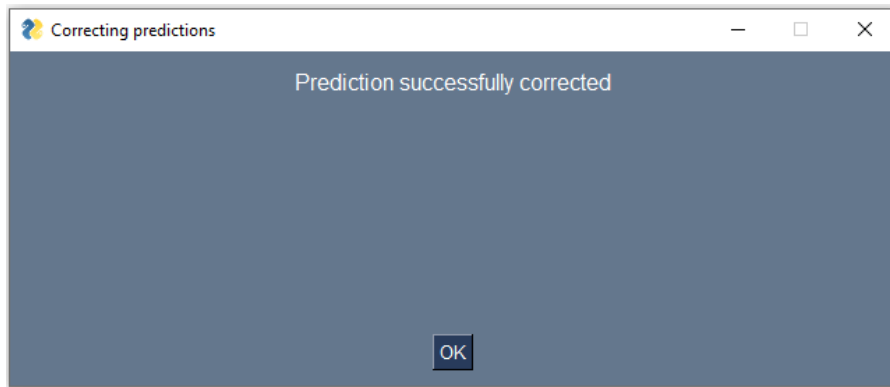


Figure 153 Message after correcting the prediction

10.3.2.2 Training model with new data

As to training any of the models, the user may select the Train model option, which will display a new dialog.

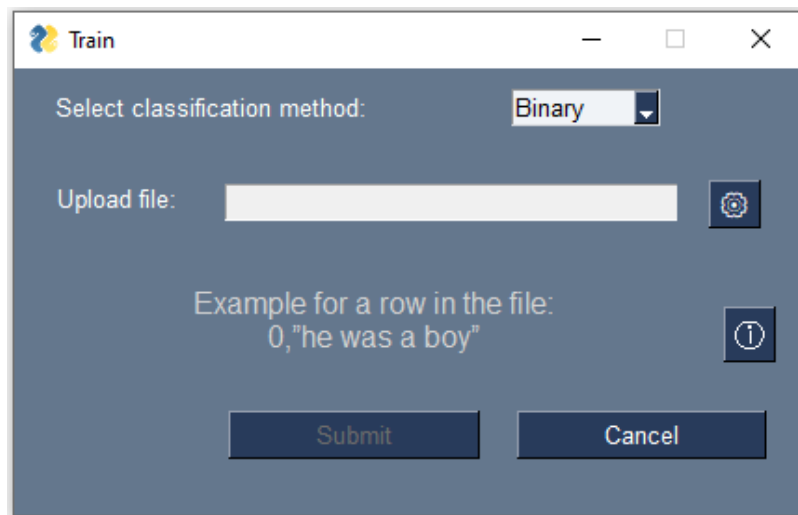


Figure 154 Train model window

In this dialog, select the model wanted to be trained and upload the file containing the new data. If the user wants, he can abort the operation with the Cancel button. Some help is given with a message explaining how the file must display its rows. When the Submit button is pressed, a new confirmation dialog will appear. It will show an operation summary and options to confirm and cancel the task.

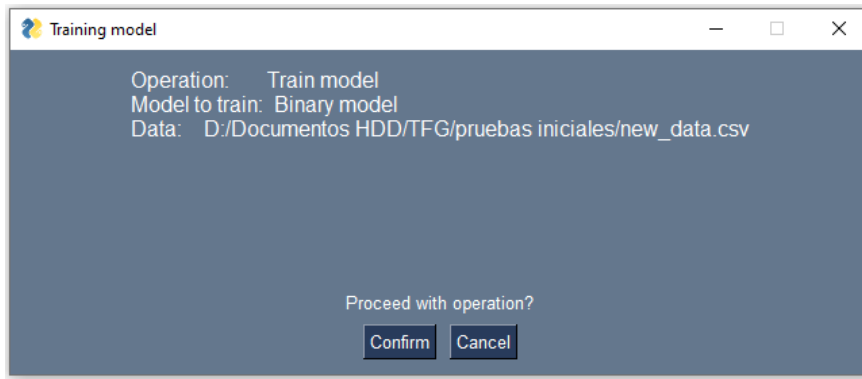


Figure 155 Confirmation window for training a model

After confirming the operation, the window may look as Figure 156, depending on the quantity of input data. **Do not close the new dialog** since the system is performing the training task.

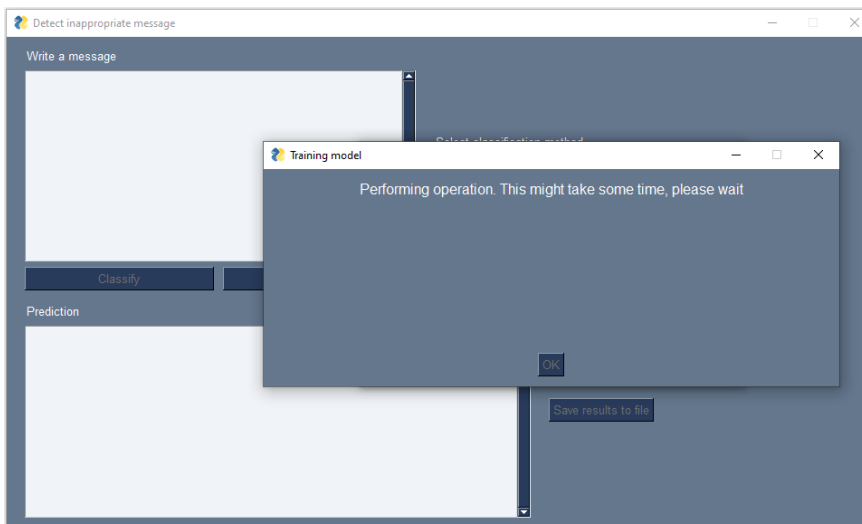


Figure 156 Confirmation window during training

After the operation ends, the dialog will show the success of the operation. If any row of the file contained incorrect or invalid data, the system automatically discards that piece of data, and informs the user after the operations finalizes.

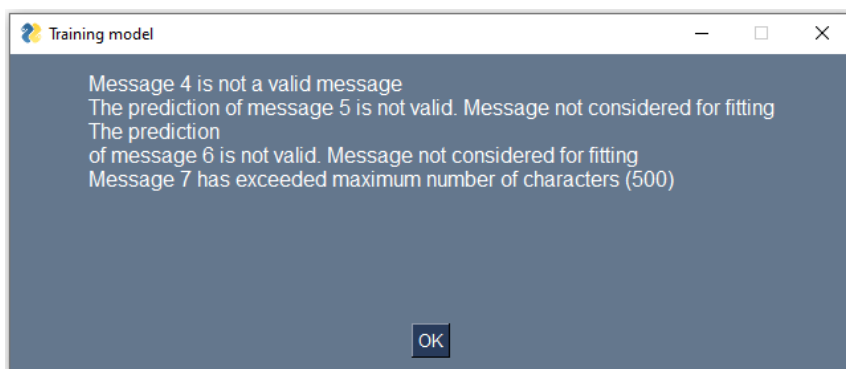


Figure 157 Finalized training with invalid messages

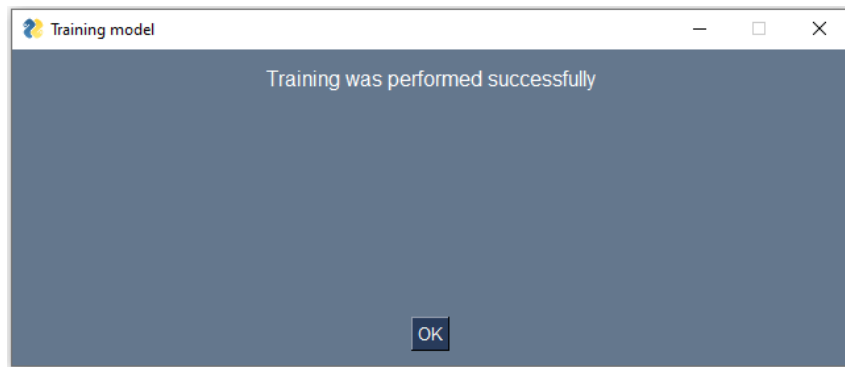


Figure 158 Finalized successful training

10.3.2.2.1 File format

This list is available on the option with the ⓘ icon. To perform this task, the file containing the unseen data must accomplish specific features.

- The file must have .csv extension.
- The file must contain only the message and the prediction values.
- The message must be enclosed between double quotes ("").
- The file must contain columns separated by commas. The number of columns depends on the model to be trained.
 - The binary model receives one column for the message and another column for the value of the prediction. This value must be 0 or 1.
 - The itemized model receives one column for the message and another six for the targets. The target values must be 0 or 1.
- The format for each row must be the following:
 - For the binary model: <prediction_value>,"<message>". For instance: 0,"he was a boy".
 - For the itemized model: "<message>",<1>,<2>,<3>,<4>,<5>,<6>:
 - 1: Toxic value.
 - 2: Severe toxic value.
 - 3: Obscene value.
 - 4: Threat value.
 - 5: Insult value.
 - 6: Identity hate value.
 - For instance: "this is a message",0,0,0,0,0,0.

10.4 Programmer manual

This subsection will suggest the methodology relative to the maintainability of the system, and how some of the changes should be made.

10.4.1 Add a new classifier

The developer in charge of creating a new classifier for the system must implement the interface `Classifier`, and have the following methods:

- `getModelOpt () → String`: For identifying the model in the user interface.
- `predict (String, int) → Prediction`: Receives the message and the number of message and returns the values of the prediction inside a `Prediction` object.
- `fitNewData (Data) → List<String>`: Receives the data and makes the necessary changes to the model to fit this new data. The list returned is the list of possible errors.
- `fitPrediction (Prediction) → List<String>`: It receives a `Prediction` object to be fitted and returns the list of possible errors.
- `toPrediction (String, int, List<int>) → Prediction`: Converts the input parameters into a `Prediction` object.

In most of the cases it would be mandatory to implement another `Prediction` class. If the model shares the same name and number of labels, the already `Prediction` classes may be used. To implement a `Prediction` class the following methods are required:

- `getMessageForUI () → String`: returns a user-friendly `String` representation of the message of the prediction.
- `getPredictionForUI () → String`: returns a user-friendly `String` representation of the prediction.
- `getPredictionForTxt () → String`: returns a `String` formatted to be the output of the prediction in the `.txt` file.
- `getHeaderForOutputFile () → List<String>`: For the output file when saving the results, returns the header first line of the new file.
- `constructPredictionForOutputFile () → List<Object>`: For the output file when saving the results, returns the row representing the prediction.

After implementing a `Classifier` and `Prediction` class, the user only has to alter the `_model_opt_to_model(model_opt)` and `_refresh_model()` to consider the new model.

To modify the interface the developer would have to do some changes in the `handle_n_msg_combo_event(window, values)` and `handle_submit_correct_pred_event(window, values)` methods, and display the new model option in the Select classification method combo box.

10.4.2 Change the interface

The developer may be responsible of integrating the system to a new user interface. In this case, this new interface should blend with the controller, who acts as an interface between the user interface and the rest of the application. These interface methods are:

- `predict(List<String>) → List<Prediction>, List<String>`: This method receives the list of messages to be predicted and returns the list of predictions and a list of errors.
- `predictMessagesInFile(String) → List<Prediction>, List<String>`: Receives the path to the messages file and returns the list of predictions and a list of errors.
- `redoLastPrediction() → List<Prediction>, List<String>`: Performs once again the predictions stored in `last_predictions`. It will not compute previous invalid predictions.
- `changeClassificationMethod(String) → List<String>`: Receives the model option represented as `String` and changes the method. Returns a list of errors.
- `authenticate(String, String) → List<String>`: Receives the input username and password and returns a list of errors.
- `correctPredictions(int, List<int>) → List<String>`: Receives the number of the message and the new values of the prediction as a list. Returns a list with errors.
- `trainModels(String, String) → List<String>`: Receives the model option represented as a `String` and the path to the file with the new data. Returns a list of errors.
- `saveResultsToCsv(String path) → List<String>`: Saves into a `.csv` file the last performed predictions. The parameter represents the destination. A list of errors is returned.
- `saveResultsToTxt(String path) → List<String>`: Saves into a `.txt` file the last performed predictions with a user-friendly format. The parameter represents the destination. A list of errors is returned.
- `clearClassification()`: Empties the attribute containing the last predictions.

Chapter 11. Conclusions and future work

This chapter summarizes the overall perception of the fulfilled project and details some possible improvements.

11.1 Conclusions

The result of this project is a system based on artificial intelligence and machine learning with the principles of extensibility and maintainability, in a management and planification framework, throughout research and analysis of the state-of-art solutions and proposed alternatives. All the knowledge acquired throughout the degree was put into test in the making of this project.

When comparing our system to the commented articles and papers, we can establish some afterthoughts. For instance, our binary model performs better than the logistic regression model in [1] at first glance. Also, it works even or better than the convolutional network proposed in [3] or the models proposed by the several teams in [4]. For the multilabel model, we could compare it with [74], and we can see that the paper proposed better performing models. This is due to the better capability of the GRU [57] neural network proposed. However, the system offers the possibility of using with a user interface both binary and multilabel models and of retraining or correcting predictions performed by them. Overall, I think I had offered a solid scalable system that may be the baseline for future work.

I have found the fulfilment of this project laborious. All the parts through which an application passes were completely developed by me, from outlining which features would be interesting to a program of this nature to the analysis, design, implementation, and testing of them. The system was created from little previous artificial intelligence knowledge. Also, the number of issues obstructing the progress of the project were numerous, and some of them inexorable.

I have learnt to clearly differentiate between machine and deep learning. I have understood how difficult the task of training a model is, and in the case of this specific text classification task, how many variables are involved when making a good text processing algorithm. For the first time in my software career, I have been limited to the capability of the machines I have within my reach, due to deep learning models being high resource-demanding. I have gained experience on evaluating when a model is performing properly or poorly. Last, I have learnt the different methods for transforming data into structures that are suitable for the machine.

The completion of this project makes me have mixed-up feelings. On one hand I have being quite erratic when it came to working on this project. The first months were tedious and low-motivating, since researching and looking up information is not quite appealing to me. The “raw” programming part is the one I look up to. Provided a problem find the optimal solution in terms of execution time, readability, extensibility, maintainability, etc, and through trial-and-error

process reach that solution. On the other hand, I have a sense of fulfilment, and some relief, that I did have completed this project without leaving big concerns aside. I have done most of the things I wanted to do.

Overall, I have gained experienced on the fields I have worked on that I am sure will be of utility in my professional career.

11.2 Extensions

Some possible improvements of the system are listed now:

- A complex deep learning model to detect hard ambiguous messages. An artificial neural network trained in several epochs would eventually know how to classify these ambiguous messages with more precision.
- A nicer user interface with a fashionable style and clearer mechanisms like a loading animation when performing administrator operations. A user interface with stylish elements may be more appealing and so the system would have a bigger reach.
- A deployment of the system in the web, so any user worldwide can access it using only the browser. Web applications are consistently growing in contrast to the extinction of desktop applications, since they are more accessible, they do not have to be downloaded to be used and the tools to implement web interfaces integrate well with web programming languages.
- A multilingual model, to detect messages from a provided language. Given we had enough training datasets, we could eventually predict a message selecting its language. This feature in addition to the web page extension would have a global outreach.
- Research how a regression model would perform this task. Regression models offer the possibility of obtaining a continuous value instead of a discrete one. This would, theoretically, mean more accurate predictions, given that the obtained result would have more precision differentiating it from other results.

Chapter 12. Project planning and final budgets

Project planning and budget on the final phase of its fulfilment are described in this chapter.

12.1 Final planning

The final planning of the project fulfilment is presented in this chapter. The final duration of this project was 8 months, from November 22nd, 2021, to Monday, 11th 2022.

ID	Task Name	Start	Finish	Resource Names
1	Project	Mon 22/11/21	Mon 11/07/22	
2	Research	Mon 22/11/21	Thu 03/02/22	
3	Related papers	Mon 22/11/21	Fri 07/01/22	Team Leader
4	Implementation alternatives	Mon 10/01/22	Fri 21/01/22	Team Leader
5	Implementation tools	Mon 24/01/22	Thu 03/02/22	Team Leader
6	Development	Fri 04/02/22	Tue 05/07/22	
7	System 1 Alternative	Fri 04/02/22	Fri 29/04/22	
8	Analysis	Fri 04/02/22	Wed 09/02/22	
9	System definition	Fri 04/02/22	Mon 07/02/22	Analyst
10	Elicitation	Tue 08/02/22	Wed 09/02/22	Analyst
11	Design	Thu 10/02/22	Thu 17/02/22	
12	Architecture design	Thu 10/02/22	Mon 14/02/22	Software Engineer
13	Diagrams and models design	Tue 15/02/22	Thu 17/02/22	Software Engineer
14	Development	Fri 18/02/22	Mon 25/04/22	
15	Implementation	Fri 18/02/22	Mon 25/04/22	
16	NLP	Fri 18/02/22	Thu 03/03/22	Senior Programmer
17	Machine learning	Fri 04/03/22	Mon 25/04/22	Senior Programmer
18	Testing	Tue 26/04/22	Fri 29/04/22	
19	Unit testing	Tue 26/04/22	Wed 27/04/22	Tester
20	Acceptance testing	Thu 28/04/22	Fri 29/04/22	Tester
21	System 2 Alternative	Fri 04/02/22	Fri 03/06/22	
22	Analysis	Fri 04/02/22	Wed 09/02/22	
23	System definition	Fri 04/02/22	Mon 07/02/22	Analyst
24	Elicitation	Tue 08/02/22	Wed 09/02/22	Analyst
25	Design	Thu 10/02/22	Thu 17/02/22	
26	Architecture design	Thu 10/02/22	Mon 14/02/22	Software Engineer
27	Diagrams and models design	Tue 15/02/22	Thu 17/02/22	Software Engineer
28	Development	Fri 04/03/22	Mon 30/05/22	
29	Implementation	Fri 04/03/22	Mon 30/05/22	
30	Machine learning	Fri 04/03/22	Mon 30/05/22	Senior Programmer
31	Testing	Tue 31/05/22	Fri 03/06/22	
32	Unit testing	Tue 31/05/22	Wed 01/06/22	Tester
33	Acceptance testing	Thu 02/06/22	Fri 03/06/22	Tester
34	Systems integration	Mon 06/06/22	Thu 23/06/22	
35	Authentication	Fri 24/06/22	Fri 24/06/22	Senior Programmer
36	Interface	Mon 27/06/22	Tue 05/07/22	Senior Programmer
37	Documentation	Thu 25/11/21	Mon 11/07/22	
38	Planning and budget	Fri 04/02/22	Mon 07/02/22	Team Leader
39	Report and introduction	Tue 08/02/22	Tue 08/02/22	Team Leader
40	Theoretical aspects	Tue 08/02/22	Wed 09/02/22	Team Leader
41	Analysis	Thu 10/02/22	Tue 15/02/22	Analyst
42	Design	Fri 18/02/22	Wed 23/02/22	Software Engineer

43	Implementation	Wed 06/07/22	Fri 08/07/22	Software Engineer
44	Testing	Mon 06/06/22	Thu 09/06/22	Tester
45	Annexes	Mon 11/07/22	Mon 11/07/22	Software Engineer
46	Conclusions	Mon 11/07/22	Mon 11/07/22	Team Leader
47	Follow-up meetings	Thu 25/11/21	Tue 05/07/22	
48	Follow-up meetings 1	Thu 25/11/21	Thu 25/11/21	Project Leader; Team Leader
49	Follow-up meetings 2	Thu 09/12/21	Thu 09/12/21	Project Leader; Team Leader
50	Follow-up meetings 3	Thu 23/12/21	Thu 23/12/21	Project Leader; Team Leader
51	Follow-up meetings 4	Thu 03/02/22	Thu 03/02/22	Project Leader; Team Leader
52	Follow-up meetings 5	Thu 17/02/22	Thu 17/02/22	Project Leader; Team Leader
53	Follow-up meetings 6	Thu 03/03/22	Thu 03/03/22	Project Leader; Team Leader
54	Follow-up meetings 7	Thu 17/03/22	Thu 17/03/22	Project Leader; Team Leader
55	Follow-up meetings 8	Thu 31/03/22	Thu 31/03/22	Project Leader; Team Leader
56	Follow-up meetings 9	Thu 28/04/22	Thu 28/04/22	Project Leader; Team Leader
57	Follow-up meetings 10	Thu 12/05/22	Thu 12/05/22	Project Leader; Team Leader
58	Follow-up meetings 11	Thu 26/05/22	Thu 26/05/22	Project Leader; Team Leader
59	Follow-up meetings 12	Thu 09/06/22	Thu 09/06/22	Project Leader; Team Leader
60	Follow-up meetings 13	Thu 23/06/22	Thu 23/06/22	Project Leader; Team Leader
61	Follow-up meetings 14	Tue 05/07/22	Tue 05/07/22	Project Leader; Team Leader

Figure 159 Final Work Breakdown Structure

The tasks of Training and Validation from both systems have been joined with the Machine Learning tasks, since they were done almost at the same time. A new task named **System integration** was added to this planning. This task represents the application itself described in the Analysis and System design sections.

12.2 Final budget

Now the final costs budget is presented.

12.2.1 Costs budget

The costs budget has changed accordingly to the new planning alterations.

I1	I2	I3	Description	Quantity	Units
01			Research		
	001		Related papers		
		01	Team Leader	280	hours
	002		Implementation alternatives		
		01	Team Leader	80	hours
	003		Implementation tools		
		01	Team Leader	72	hours
02			Following up		
	001		Follow-up meetings		
		01	Project Leader	7	hours
		02	Team Leader	7	hours

Figure 160 Final Costs Budget Research and following Item 1

I1	I2	I3	Price	Subtotal (3)	Subtotal (2)	Total
01						16,416.00 €
	001				10,640.00 €	
		01	38.00 €	10,640.00 €		
	002				3,040.00 €	
		01	38.00 €	3,040.00 €		
	003				2,736.00 €	
		01	38.00 €	2,736.00 €		
02						714.00 €
	001				714.00 €	
		01	64.00 €	448.00 €		
		02	38.00 €	266.00 €		
					TOTAL	17,130.00 €

Figure 161 Final Costs Budget Research and following Item 2

I1	I2	I3	I4	Description	Quantity	Units
01				System 1 Alternative		
	001			Analysis		
		0001		System definition		
			01	Analyst	16	hours
		0002		Elicitation		
			01	Analyst	16	hours
	002			Design		
		0001		Architecture design		
			01	Software Engineer	24	hours
		0002		Diagrams and models design		
			01	Software Engineer	24	hours
	003			Development		
		0001		Implementation		
			01	Senior Programmer	376	hours
	004			Testing		
		0001		Unit testing		
			01	Tester	16	hours
		0002		Acceptance testing		
			01	Tester	16	hours
02				System 2 Alternative		
	001			Analysis		
		0001		System definition		
			01	Analyst	16	hours

		0002		Elicitation		
			01	Analyst	16	hours
	002			Design		
		0001		Architecture design		
			01	Software Engineer	24	hours
		0002		Diagrams and models design		
			01	Software Engineer	24	hours
	003			Development		
		0001		Implementation		
			01	Senior Programmer	496	hours
	004			Testing		
		0001		Unit testing		
			01	Tester	16	hours
		0002		Acceptance testing		
			01	Tester	16	hours
	03			System integration		
			01	Senior Programmer	112	hours
	04			Authentication		
			01	Senior Programmer	8	hours
	05			Interface		
			01	Senior Programmer	56	hours

Figure 162 Final Costs Budget Development Item 1

I1	I2	I3	I4	Price	Subtotal (4)	Subtotal (3)	Subtotal (2)	Total
01								11,168.00 €
	001						848.00 €	
		0001				424.00 €		
			01	26.50 €	424.00 €			
		0002				424.00 €		
			01	26.50 €	424.00 €			
	002						1,344.00 €	
		0001				672.00 €		
			01	28.00 €	672.00 €			
		0002				672.00 €		
			01	28.00 €	672.00 €			

003						8,272.00 €	
	0001					8,272.00 €	
		01	22.00 €	8,272.00 €			
004						704.00 €	
	0001					352.00 €	
		01	22.00 €	352.00 €			
	0002					352.00 €	
		01	22.00 €	352.00 €			
02							13,808.00 €
	001					848.00 €	
	0001					424.00 €	
		01	26.50 €	424.00 €			
	0002					424.00 €	
		01	26.50 €	424.00 €			
	002					1,344.00 €	
	0001					672.00 €	
		01	28.00 €	672.00 €			
	0002					672.00 €	
		01	28.00 €	672.00 €			
	003					10,912.00 €	
	0001					10,912.00 €	
		01	22.00 €	10,912.00 €			
	004					704.00 €	
	0001					352.00 €	
		01	22.00 €	352.00 €			
	0002					352.00 €	
		01	22.00 €	352.00 €			
03							2,464.00 €
		01	22.00 €	2,464.00 €			
04							176.00 €
		01	22.00 €	176.00 €			
05							1,232.00 €
		01	22.00 €	1,232.00 €			
						TOTAL	26,384.00 €

Figure 163 Final Costs Budget Development Item 2

I1	I2	Description	Quantity	Units
01		Report and introduction		
		Team Leader	8	hours
02		Theoretical aspects		
	01	Team Leader	16	hours
03		Planning and budget		
	01	Team Leader	16	hours
04		Analysis		
	01	Analyst	32	hours
05		Design		
	01	Software Engineer	32	hours
06		Implementation		
	01	Software Engineer	24	hours
07		Testing		
	01	Tester	32	hours
08		Conclusions		
		Project Leader	8	hours
09		Annexes		
	01	Team Leader	8	hours

Figure 164 Final Costs Budget Documentation Item 1

I1	I2	Price	Subtotal (2)	Total
01				304.00 €
		38.00 €	304.00 €	
02				608.00 €
	01	38.00 €	608.00 €	
03				608.00 €
	01	38.00 €	608.00 €	
04				848.00 €
	01	26.50 €	848.00 €	
05				896.00 €
	01	28.00 €	896.00 €	
06				672.00 €
	01	28.00 €	672.00 €	
07				704.00 €
	01	22.00 €	704.00 €	
08				512.00 €
		64.00 €	512.00 €	
09				304.00 €
	01	38.00 €	304.00 €	

			TOTAL	5,152.00 €

Figure 165 Final Costs Budget Documentation Item 2

This is the final costs budget summary.

Item	Item name	Total
01	Research and following up	17,130.00 €
02	Development	26,384.00 €
03	Documentation	5,152.00 €
	Total Cost	48,666.00 €

Figure 166 Final Costs Budget summary

Chapter 13. Bibliographic references

- [1] T. Davidson, D. Warmley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," Mar. 2017. Accessed: Jul. 03, 2022. [Online]. Available: <https://arxiv.org/pdf/1703.04009.pdf>
- [2] E. Spertus, "Smokey- Automatic Recognition of Hostile Messages," 1997. Accessed: Jul. 03, 2022. [Online]. Available: <https://www.aaai.org/Papers/IAAI/1997/IAAI97-209.pdf>
- [3] N. Banik and H. H. Rahman, "Toxicity Detection on Bengali Social Media Comments using Supervised Models," Nov. 2019. Accessed: Jul. 03, 2022. [Online]. Available: https://www.researchgate.net/publication/337317824_Toxicity_Detection_on_Bengali_Social_Media_Comments_using_Supervised_Models
- [4] M. Ezra Aragon, M. A. Alvarez Carmona, M. Montes, and H. Jair Escalante, "Overview of MEX-A3T at IberLEF 2019: Authorship and aggressiveness analysis in Mexican Spanish tweets," Aug. 2019. Accessed: Jul. 03, 2022. [Online]. Available: https://www.researchgate.net/publication/334973555_Overview_of_MEX-A3T_at_IberLEF_2019_Authorship_and_aggressiveness_analysis_in_Mexican_Spanish_tweets
- [5] H. Sohn and H. Lee, "MC-BERT4HATE: Hate speech detection using multi-channel bert for different languages and translations," IEEE Computer Society, Nov. 2019. doi: 10.1109/ICDMW.2019.00084.
- [6] IBM, "What is Logistic regression? | IBM." <https://www.ibm.com/topics/logistic-regression> (accessed Jul. 04, 2022).
- [7] D. Krishna, "A Look at the Maths Behind Linear Classification | by Dhruva Krishna | Towards Data Science," Dec. 23, 2020. <https://towardsdatascience.com/a-look-at-the-maths-behind-linear-classification-166e99a9e5fb> (accessed Jul. 04, 2022).
- [8] IBM, "What are Neural Networks? | IBM." <https://www.ibm.com/cloud/learn/neural-networks> (accessed Jul. 04, 2022).
- [9] J. Brownlee, "One-vs-Rest and One-vs-One for Multi-Class Classification," Apr. 27, 2020. <https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/> (accessed Jul. 07, 2022).
- [10] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification."
- [11] JCharisTech, "Multi-Label Text Classification with Scikit-MultiLearn in Python | YouTube," Sep. 15, 2020. <https://www.youtube.com/watch?v=YyOuDi-zSiI&t=283s> (accessed Jul. 10, 2022).

- [12] J. McCarthy, "What is Artificial Intelligence?," Nov. 2004. Accessed: Jul. 03, 2022. [Online]. Available: https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatissai.pdf
- [13] A. Turing, "Computing Machinery and Intelligence," 1950. Accessed: Jul. 03, 2022. [Online]. Available: <https://www.csee.umbc.edu/courses/471/papers/turing.pdf>
- [14] R. Anyoha, "The History of Artificial Intelligence - Science in the News," Aug. 28, 2017. <https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/> (accessed Jul. 03, 2022).
- [15] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," Jul. 1959.
- [16] P. Abbeel, "Step by Step- Alpha Beta Pruning - YouTube," Feb. 09, 2013. <https://www.youtube.com/watch?v=xBXHz4Gbdo> (accessed Jul. 03, 2022).
- [17] P. Vadapalli, "Min Max Algorithm in AI: Components, Properties, Advantages & Limitations | upGrad blog," Dec. 22, 2020. <https://www.upgrad.com/blog/min-max-algorithm-in-ai/> (accessed Jul. 03, 2022).
- [18] K. D. Foote, "A Brief History of Machine Learning - DATAVERSITY," Dec. 03, 2021. <https://www.dataversity.net/a-brief-history-of-machine-learning/> (accessed Jul. 03, 2022).
- [19] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity - SpringerLink," Dec. 1943. Accessed: Jul. 03, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/BF02478259>
- [20] IBM, "About Linear Regression | IBM." <https://www.ibm.com/topics/linear-regression> (accessed Jul. 04, 2022).
- [21] J. Brownlee, "Linear Regression for Machine Learning," Aug. 15, 2020. <https://machinelearningmastery.com/linear-regression-for-machine-learning/> (accessed Jul. 04, 2022).
- [22] C. (Dotcsv) Santana Vega, "Regresión Lineal y Mínimos Cuadrados Ordinarios | YouTube," Dec. 16, 2017. https://www.youtube.com/watch?v=k964_uNn3l0 (accessed Jul. 04, 2022).
- [23] S. Sekhar, "Math Behind Logistic Regression Algorithm," Aug. 20, 2019. <https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c> (accessed Jul. 04, 2022).
- [24] P. Gupta, "Regularization in Machine Learning | Towards Data Science," Nov. 15, 2017. <https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a> (accessed Jul. 05, 2022).

- [25] A. v Srinivasan, “Stochastic Gradient Descent — Clearly Explained !! | Towards Data Science,” Sep. 07, 2019. <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31> (accessed Jul. 05, 2022).
- [26] Scikit Learn, “Stochastic Gradient Descent — scikit-learn.” <https://scikit-learn.org/stable/modules/sgd.html> (accessed Jul. 05, 2022).
- [27] GeekForGeeks, “Stochastic Gradient Descent (SGD) | GeeksforGeeks,” Jun. 13, 2022. <https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/> (accessed Jul. 08, 2022).
- [28] C. Santana Vega, “¿Qué es una Red Neuronal? Parte 1 : La Neurona | YouTube,” Mar. 19, 2018. <https://www.youtube.com/watch?v=MRlv2lwFTPg> (accessed Jul. 04, 2022).
- [29] B. Muller, “BERT 101 - State Of The Art NLP Model Explained,” Mar. 02, 2022. <https://huggingface.co/blog/bert-101> (accessed Jul. 04, 2022).
- [30] R. Horev, “BERT Explained: State of the art language model for NLP | Towards Data Science,” Nov. 10, 2018. <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> (accessed Jul. 04, 2022).
- [31] B. Veners, “The Making of Python,” Jan. 13, 2003. <https://www.artima.com/articles/the-making-of-python> (accessed Jul. 04, 2022).
- [32] “scikit-learn: machine learning in Python.” <https://scikit-learn.org/stable/> (accessed Jul. 04, 2022).
- [33] “PyTorch.” <https://pytorch.org/> (accessed Jul. 04, 2022).
- [34] “TensorFlow.” <https://www.tensorflow.org/?hl=es-419> (accessed Jul. 04, 2022).
- [35] “JAX documentation.” <https://jax.readthedocs.io/en/latest/notebooks/quickstart.html> (accessed Jul. 04, 2022).
- [36] Refactoring.Guru, “Strategy pattern.” <https://refactoring.guru/es/design-patterns/strategy> (accessed Jul. 03, 2022).
- [37] “pickle — Python object serialization.” <https://docs.python.org/3/library/pickle.html> (accessed Jul. 10, 2022).
- [38] SQLite, “SQLite Home Page.” <https://www.sqlite.org/index.html> (accessed Jul. 03, 2022).
- [39] Yale University, “WCAG 2 A and AA Checklist | Usability & Web Accessibility.” <https://usability.yale.edu/web-accessibility/articles/wcag2-checklist> (accessed Jul. 05, 2022).
- [40] A. U. Iyer, “Toxic Tweets Dataset | Kaggle.” <https://www.kaggle.com/datasets/ashwiniyer176/toxic-tweets-dataset> (accessed Jul. 03, 2022).

- [41] L. Dixon, J. Sorensen, and nithum, "Toxic Comment Classification Challenge | Kaggle," 2018. <https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data> (accessed Jul. 03, 2022).
- [42] "Python Release Python 3.10.2 | Python.org." <https://www.python.org/downloads/release/python-3102/> (accessed Jul. 10, 2022).
- [43] "Why is Python So Popular? | Pulumu." <https://www.pulumu.com/why-is-python-so-popular/> (accessed Jul. 05, 2022).
- [44] aigamedev, "scikit-neuralnetwork," 2016. <https://github.com/aigamedev/scikit-neuralnetwork> (accessed Jul. 05, 2022).
- [45] "Keras: the Python deep learning API." <https://keras.io/> (accessed Jul. 05, 2022).
- [46] "tkinter." <https://docs.python.org/es/3/library/tkinter.html> (accessed Jul. 03, 2022).
- [47] "Qt for Python." <https://doc.qt.io/qtforpython/> (accessed Jul. 03, 2022).
- [48] "Python REMote Interface library. Platform independent. In about 100 Kbytes, perfect for your diet." <https://github.com/rawpython/remi> (accessed Jul. 03, 2022).
- [49] "Welcome to wxPython! | wxPython." <https://www.wxpython.org/> (accessed Jul. 03, 2022).
- [50] Microsoft, "Visual Studio Code - Code Editing." <https://code.visualstudio.com/> (accessed Jul. 05, 2022).
- [51] "Git." <https://git-scm.com/> (accessed Jul. 05, 2022).
- [52] J. Brownlee, "A Gentle Introduction to the Bag-of-Words Model," 2017. <https://machinelearningmastery.com/gentle-introduction-bag-words-model/> (accessed Jul. 03, 2022).
- [53] Scikit Learn, "sklearn.multioutput.MultiOutputClassifier | scikit-learn." <https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputClassifier.html> (accessed Jul. 06, 2022).
- [54] L. Ramadhan, "TF-IDF Simplified | Towards Data Science," 2021. <https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530> (accessed Jul. 03, 2022).
- [55] anokas, "Discussion: Keras sample weight for imbalance multilabel datasets | Toxic Comment Classification Challenge | Kaggle," 2017. <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/46673> (accessed Jul. 03, 2022).
- [56] NVIDIA, "CUDA Zone - Library of Resources | NVIDIA Developer." <https://developer.nvidia.com/cuda-zone> (accessed Jul. 03, 2022).

- [57] S. Kostadinov, "Understanding GRU Networks | Towards Data Science," 2017. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> (accessed Jul. 03, 2022).
- [58] R. Dolphin, "LSTM Networks | A Detailed Explanation | Towards Data Science," 2020. <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9> (accessed Jul. 03, 2022).
- [59] jonad, "jonad/Toxicity_comments," 2020. https://github.com/jonad/Toxicity_comments (accessed Jul. 03, 2022).
- [60] D. Karani, "Introduction to Word Embedding and Word2Vec | Towards Data Science," Sep. 01, 2018. <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa> (accessed Jul. 03, 2022).
- [61] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation." <https://nlp.stanford.edu/projects/glove/> (accessed Jul. 03, 2022).
- [62] Facebook AI Reseach, "fastText." <https://fasttext.cc/> (accessed Jul. 03, 2022).
- [63] J. Martinez Heras, "Precision, Recall, F1, Accuracy en clasificación - IArtificial.net," Oct. 09, 2020. <https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/> (accessed Jul. 03, 2022).
- [64] tutorialspoint, "What is Hamming Distance?" <https://www.tutorialspoint.com/what-is-hamming-distance> (accessed Jul. 03, 2022).
- [65] "pydoc — Generador de documentación y Sistema de ayuda en línea — documentación de Python - 3.10.5." <https://docs.python.org/es/3/library/pydoc.html> (accessed Jul. 09, 2022).
- [66] A. Paraschiv, "Detecting Toxic Behavior in Social Media and Online News," Jun. 2020. Accessed: Jul. 04, 2022. [Online]. Available: https://www.researchgate.net/publication/353481771_Detecting_Toxic_Behavior_in_Social_Media_and_Online_News
- [67] A. Kumar, V. Tyagi, and S. Das, "Deep Learning for Hate Speech Detection in social media," Institute of Electrical and Electronics Engineers Inc., Sep. 2021. doi: 10.1109/GUCON50781.2021.9573687.
- [68] A. Wisnugraha, "Classification of Toxic Tweets | Kaggle," 2021. <https://www.kaggle.com/code/adityawisnugrahas/classification-of-toxic-tweets-bigru-92-84> (accessed Jul. 04, 2022).
- [69] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks | Towards Data Science," Dec. 15, 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (accessed Jul. 05, 2022).

- [70] L. Shi, K. Du, C. Zhang, H. Ma, and W. Yan, "Lung Sound Recognition Algorithm Based on VGGish-BiGRU," *IEEE Access*, vol. 7, pp. 139438–139449, 2019, doi: 10.1109/ACCESS.2019.2943492.
- [71] neongen, "2nd place solution overview | Toxic Comment Classification Challenge | Kaggle," 2018. <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52630> (accessed Jul. 05, 2022).
- [72] A. Burmistrov, "About my 0.9872 single model | Toxic Comment Classification Challenge | Kaggle," 2018. <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52644> (accessed Jul. 05, 2022).
- [73] M. M. Kazanova, "5th place Brief Solution | Toxic Comment Classification Challenge | Kaggle," 2018. <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52612> (accessed Jul. 05, 2022).
- [74] S. Li, "Application of Recurrent Neural Networks In Toxic Comment Classification," 2018. Accessed: Jul. 04, 2022. [Online]. Available: <https://escholarship.org/uc/item/5f87h061>
- [75] N. Hubens, "Test Time Augmentation (TTA) and how to perform it with Keras | Towards Data Science," Feb. 11, 2019. <https://towardsdatascience.com/test-time-augmentation-tta-and-how-to-perform-it-with-keras-4ac19b67fb4d> (accessed Jul. 05, 2022).
- [76] S. Narkhede, "Understanding AUC - ROC Curve | Towards Data Science," Jun. 26, 2018. <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5> (accessed Jul. 05, 2022).
- [77] Clement and Yahya, "Logistic regression python solvers' definitions - Stack Overflow," 2021. <https://stackoverflow.com/questions/38640109/logistic-regression-python-solvers-definitions> (accessed Jul. 05, 2022).
- [78] S. Sharma, "What the Hell is Perceptron?. The Fundamentals of Neural Networks | Towards Data Science," Sep. 09, 2017. <https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53> (accessed Jul. 05, 2022).
- [79] Santana Vega C, "¿Qué es el Descenso del Gradiente? Algoritmo de Inteligencia Artificial | YouTube," Feb. 04, 2018. https://www.youtube.com/watch?v=A6FiCDoz8_4 (accessed Jul. 05, 2022).
- [80] eldrich, "Hate speech offensive tweets by Davidson et al | Kaggle," 2020. <https://www.kaggle.com/datasets/eldrich/hate-speech-offensive-tweets-by-davidson-et-al> (accessed Jul. 05, 2022).
- [81] nicapotato, "Bad Bad Words | Kaggle," 2018. <https://www.kaggle.com/datasets/nicapotato/bad-bad-words> (accessed Jul. 05, 2022).
- [82] A. Pai, "What is Tokenization | Tokenization In NLP," May 26, 2020. <https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/> (accessed Jul. 05, 2022).

- [83] P. Sharma, "An Introduction to Stemming in Natural Language Processing," Nov. 25, 2021. <https://www.analyticsvidhya.com/blog/2021/11/an-introduction-to-stemming-in-natural-language-processing/> (accessed Jul. 05, 2022).
- [84] S. Srinidhi, "Understanding Word N-grams and N-gram Probability in Natural Language Processing | Towards Data Science," Nov. 27, 2019. <https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058> (accessed Jul. 05, 2022).
- [85] J. Nabi, "Machine Learning — Multiclass Classification with Imbalanced Dataset | Towards Data Science," Dec. 22, 2018. <https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a> (accessed Jul. 06, 2022).
- [86] Scikit Learn, "sklearn.linear_model.LogisticRegression | scikit-learn." https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed Jul. 05, 2022).
- [87] G. Shperber, "A gentle introduction to Doc2Vec. TL;DR | Medium," Jul. 26, 2017. <https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e> (accessed Jul. 05, 2022).
- [88] Zach, "A Simple Explanation of the Jaccard Similarity Index | Statology," Dec. 23, 2020. <https://www.statology.org/jaccard-similarity/> (accessed Jul. 07, 2022).
- [89] J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks," Dec. 03, 2018. <https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/> (accessed Jul. 07, 2022).
- [90] "PyTorch Linear Layer (Fully Connected Layer) Explained," Feb. 02, 2022. <https://androidkt.com/pytorch-linear-layer-fully-connected-layer-explained/> (accessed Jul. 07, 2022).
- [91] "13.12. Chapter Summary | Attention Layers." <https://dmol.pub/dl/attention.html> (accessed Jul. 07, 2022).
- [92] HuggingFace, "Tokenizer." https://huggingface.co/docs/transformers/main_classes/tokenizer (accessed Jul. 07, 2022).
- [93] "scikit-multilearn: Multi-Label Classification in Python — Multi-Label Classification for Python." <http://scikit.ml/> (accessed Jul. 08, 2022).
- [94] Scikit Learn, "sklearn.model_selection.GridSearchCV — scikit-learn." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (accessed Jul. 07, 2022).
- [95] "Contrast Ratio: Easily calculate color contrast ratios. Passing WCAG was never this easy!" <https://contrast-ratio.com/#white-on-%2364778D> (accessed Jul. 08, 2022).

- [96] W3C, "Understanding Success Criterion 1.3.3: Sensory Characteristics." <https://www.w3.org/WAI/WCAG21/Understanding/sensory-characteristics> (accessed Jul. 05, 2022).
- [97] W3C, "Understanding Success Criterion 1.4.1: Use of Color." <https://www.w3.org/WAI/WCAG21/Understanding/use-of-color> (accessed Jul. 05, 2022).
- [98] W3C, "Understanding Success Criterion 1.4.3: Contrast (Minimum)." <https://www.w3.org/WAI/WCAG21/Understanding/contrast-minimum> (accessed Jul. 08, 2022).
- [99] W3C, "Understanding Success Criterion 1.4.12: Text Spacing." <https://www.w3.org/WAI/WCAG21/Understanding/text-spacing> (accessed Jul. 05, 2022).
- [100] W3C, "Understanding Success Criterion 2.1.1: Keyboard." <https://www.w3.org/WAI/WCAG21/Understanding/keyboard> (accessed Jul. 05, 2022).
- [101] W3C, "Understanding Success Criterion 2.1.2: No Keyboard Trap." <https://www.w3.org/WAI/WCAG21/Understanding/no-keyboard-trap> (accessed Jul. 05, 2022).
- [102] W3C, "Understanding Success Criterion 2.2.1: Timing Adjustable." <https://www.w3.org/WAI/WCAG21/Understanding/timing-adjustable> (accessed Jul. 05, 2022).
- [103] W3C, "Understanding Success Criterion 2.2.2: Pause, Stop, Hide." <https://www.w3.org/WAI/WCAG21/Understanding/pause-stop-hide> (accessed Jul. 05, 2022).
- [104] W3C, "Understanding Success Criterion 2.4.7: Focus Visible." <https://www.w3.org/WAI/WCAG21/Understanding/focus-visible> (accessed Jul. 05, 2022).
- [105] W3C, "Understanding Success Criterion 2.5.2: Pointer Cancellation." <https://www.w3.org/WAI/WCAG21/Understanding/pointer-cancellation> (accessed Jul. 05, 2022).
- [106] W3C, "Understanding Success Criterion 2.5.3: Label in Name." <https://www.w3.org/WAI/WCAG21/Understanding/label-in-name> (accessed Jul. 05, 2022).
- [107] W3C, "Understanding Success Criterion 3.2.2: On Input." <https://www.w3.org/WAI/WCAG21/Understanding/on-input> (accessed Jul. 05, 2022).
- [108] W3C, "Understanding Success Criterion 3.3.1: Error Identification." <https://www.w3.org/WAI/WCAG21/Understanding/error-identification> (accessed Jul. 05, 2022).

- [109] W3C, “Understanding Success Criterion 3.3.2: Labels or Instructions.” <https://www.w3.org/WAI/WCAG21/Understanding/labels-or-instructions> (accessed Jul. 05, 2022).
- [110] W3C, “Understanding Success Criterion 3.3.3: Error Suggestion.” <https://www.w3.org/WAI/WCAG21/Understanding/error-suggestion> (accessed Jul. 05, 2022).

Chapter 14. Annexes

14.1 Glossary

- **Administrator.** An authenticated user. An administrator can perform tasks like training a model with unseen data and correcting a prediction performed by any classifier.
- **Appropriate.** Property of a message which is not inappropriate. See Inappropriate.
- **Accuracy.** Metric used for evaluating a model. It is calculated as $\frac{TP+TN}{TP+TN+FP+FN}$.
- **Artificial intelligence.** Ability of a machine to mimic the behaviour of the human mind on problem-solving and decision-making [12].
- **Authenticate.** Log in as an administrator.
- **Authentication.** The process of logging in as administrator.
- **BERT model.** Deep learning model developed by Google used in natural language processing tasks and prepared to be fine-tuned to a specific problem.
- **Binary model.** Model that receives an input and produces an output that may have two possible values.
- **Classification (machine learning).** Field inside machine learning in which a model tries to predict a discrete value from a set of unseen data.
- **Classification (prediction).** See prediction.
- **Classifier.** Algorithm that manages classification tasks. See Classification (machine learning).
- **Confusion matrix.** Metric used for evaluating a model performance. It consists of a data structure that stores the instances of correct and incorrect predictions with respect to the true ones.
- **Deep learning.** Field inside machine learning where systems developed try to imitate the structure of the human brain. Its foundation lies in the neural networks of the brain.
- **False negative/FN.** The number of instances that were predicted as negative but were positive.
- **False positive/FP.** The number of instances that were predicted as positive but were negative.
- **Feature (classifier).** Set of data provided to a classifier to be predicted.
- **F1 Score.** Metrics used for evaluating a model. It is calculated as $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.
- **Inappropriate.** Property socially conferred to whichever message that exhibits elements of toxicity, obscenity, threat, insult and/or hate speech within its meaning.
- **Itemized model.** Model that receives an input and produces a fixed number of outputs that may have two possible values.
- **Jigsaw competition.** Competition held in Kaggle in 2018 regarding toxic comment classification.
- **Label.** Independent variable that will have the value of a prediction.
- **Linear regression.** Mathematical model which allows to predict a dependent value with a set of independent ones.

- **Logistic regression.** Statistical model which allows to predict a categorical value given an input.
- **Machine learning.** Field inside artificial intelligence that gathers the methods and techniques for imitating human learning behaviour.
- **Model.** See Classifier.
- **Multilabel model.** See Itemized model.
- **Neural network.** Deep learning model which represents an abstraction of the human brain structure from the real world having the neuron as its atomic unit.
- **NLP.** Natural Language Processing. It involves all the algorithms to manipulate and analyse the human language-related interactions with a machine.
- **Overfitting.** Characteristic present in models that correctly classify training data but are prone-to-error with unseen data.
- **Precision.** Metric used for evaluating a model. It is calculated as $\frac{TP}{TP+FP}$.
- **Prediction.** The result of a model which has been provided a set of features.
- **Python.** High-level programming language widely used in machine and deep learning applications.
- **PyTorch.** Python library that offers deep learning modules to implement neural networks.
- **Recall.** Metric used for evaluating a model. It is calculated as $\frac{TP}{TP+FN}$.
- **Scikit Learn.** Python library that offers machine learning functionality to implement models to be fine-tuned.
- **Strategy pattern.** Design pattern widely used in software engineering that allows a context class to perform the same operation with different algorithms.
- **Target.** See Label.
- **Transformers.** Python library offered by the HuggingFace company to implement state-of-art machine learning and deep learning programs.
- **TN/True negatives.** The number of correctly predicted negative instances.
- **TP/True Positive.** The number of correctly predicted positive instances.

14.2 Delivered content in attached file

This section lists all the content attached to this document.

14.2.1 Contents

Figure 167 shows the content of the attached file.

File/Folder	Content
<i>hate-speech-detection.zip</i>	This compressed file contains the source code of the project. It also contains a

	README.txt explaining all the steps to start the application, a requirements file and a JSON file with initialization configuration.
<i>doc/</i>	This folder will contain all the documentation regarding this project. There are two subfolders inside it.
<i>doc/project/</i>	Contains the Microsoft Project file where all the planning has been set and the Microsoft Excel file with initial and final budgets.
<i>doc/pydoc/</i>	Contains the generated documentation of the source code.
<i>doc/logs/</i>	This folder contains logs from executed scripts during implementation.
<i>Adrian-Perez-Manso-hate-speech-detection.pdf</i>	The current document.
<i>file examples/</i>	This folder contains examples for all the use cases where the user may submit a file into the system.
<i>README.txt</i>	Installation and execution steps

Figure 167 Contents of the attached file

If we unzip the compressed file, we will have the following directory:

File/Folder	Content
<i>database/</i>	Folder that contains the database of users.
<i>datasets/</i>	Folder that contains the different datasets for the binary and multilabel models.
<i>help/</i>	Folder that contains some guidance about the files' format that may be submitted to the system.
<i>models/</i>	Folder that contains the serialized objects needed for the two models.
<i>src/</i>	Folder that contains the source code. Contains the <code>main.py</code> file.
<i>testfiles/</i>	Folder with example files for testing.
<i>init.json</i>	File with execution configuration. Explained in 10.2.
<i>requirements.txt</i>	File with the needed modules to be installed. Explained in 10.1.

Figure 168 Contents of the compressed file

14.2.2 Executable code and installation

We can execute the program by running Python with the `main.py` file inside the `src/` folder. If any detail is necessary, follow steps in sections 10.1 and 10.2.

14.3 Source code

The source code of the application is inside the `hate-speech-detection.zip` compressed file inside the attached content.

14.4 Meeting minutes

All the meetings with the tutor are shown in Figure 169.

Meeting	Date	Mean	Content
Meeting 1	November 11 th , 2021	Microsoft Teams	First immediate goals
Meeting 2	November 25 th , 2021	Microsoft Teams	Updates on research, planning and budget
Meeting 3	December 9 th , 2021	Microsoft Teams	Updates on research, planning and budget
Meeting 4	December 23 rd , 2021	Microsoft Teams	Updates on research, planning and budget
Meeting 5	February 3 rd , 2022	Microsoft Teams	Updates on research, planning and budget
Meeting 6	February 17 th , 2022	Microsoft Teams	First system implementations
Meeting 7	March 3 rd , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 8	March 17 th , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 9	March 31 st , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 10	April 28 th , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 11	May 12 th , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 12	May 26 th , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 13	June 9 th , 2022	Microsoft Teams	Update on system implementation and found problems
Meeting 14	June 23 rd , 2022	Microsoft Teams	Document revision and problems found
Meeting 15	July 5 th , 2022	Microsoft Teams	Document and application revision

Figure 169 Meeting minutes