APPLIED NLP AND ML FOR THE DETECTION OF INAPPROPRIATE TEXT IN A COMMUNICATION PLATFORM

AITOR URRUTIA ZUBIKARAI

Thesis Supervisor: Jordi Riera
Tutor: René Alquézar

A THESIS SUBMITTED FOR THE DEGREE OF MASTER IN ARTIFICIAL INTELLIGENCE

UNIVERSITAT POLITÈCNICA DE CATALUNYA, UNIVERSITAT DE BARCELONA & UNIVERSITAT ROVIRA I VIRGILI

January 2020

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Aitor Urrutia Zubikarai
January 2020

Acknowledgments

First of all, I wish to express my sincere appreciation to the NexTReT S.L. company for supporting this Master Thesis, to the Tellfy development team for their help and specially to Jordi Riera for being my thesis supervisor and encouraging me to develop the best work possible. As well, I would like to thank my tutor René Alquézar for guiding me in this whole process.

More importantly, I wish to acknowledge the support and great love of my family and specially to my partner Irati for supporting me in every moment because this work would not have been possible without their help.

Finally, I can not describe with words the deep sense of gratitude I feel for all your daily help during this whole year, Arnau. I am greatly thankful for all your tips and advice during this work and all I have been able to learn by your side. It has been a great year working side by side with you and I hope that some day we will be able to work together again.

Abstract

With the expansion of the communication platforms, individuals are very used to exchange information and communicate with other people through these platforms, both for social and business purposes. It's a known problem, that many people use the anonymity provided by these platforms to use inappropriate and offensive language.

The company NextreT S.L. has built a communication platform directed to business use. As the company wants to avoid offensive language in this platform, natural language processing tools in big data environment are going to be used to analyze each written text to detect and remove if required this inappropriate and offensive language.

During this project, a variety of techniques used in the State of the Art are analyzed, compared and then tested using a completely new data set in Spanish language created using Tellfy App and Twitter corpus.

Initially, different word encoding methods are tested, including word embedding like Word2Vec and FastText. In addition, different hyper parameter configurations are checked as well as model performances with different data sizes. Finally, after a forward feature selection phase, model ensemble techniques are tested.

During these tests, it has been shown that the combination of the features that are used is very important to increase the performance of the models. Also, the different word representation techniques are very related to the performance of the models. Furthermore, the sizes of the training sets that are used need to be as representative and as large as possible. Finally, after using different complex Deep Neural Network models, more traditional Logistic Regression models can offer a better performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the 21st century the importance of information technology has grown incredibly as well as the use of this technology for communication uses. From data extracted by [Chaa] by January of 2019, the amount of unique mobile users was of 5.112 billions (about a 67% of the total population in earth). The internet users were up to 4.388 billions (about a 57%) and the active social media users were 3.484 billions (45% of the populations, almost the half of the population).

Although the use of Internet is very high worldwide (more than the half of the population), in developed countries more than a 80% of the population uses Internet. For example, around a 95% of inhabitants from northern America, western Europe and northern Europe are Internet users.

Referring to the communication platforms, such as WhatsApp, Facebook Messenger, Viber, WeChat, Line, Telegram, imo and Kakaotalk, their use has increased in the last years becoming 1.5 billion users in some of them (WhatsApp and Facebook Messenger) as points [Buc].

In the same article, 14 of the most developed areas of the world are analyzed so as to see these applications usage, penetration and statistics. These regions are Great Britain, USA, India, Singapore, APAC Region, South Africa, Germany, China, Austria, Brazil, Spain, Italy, France and Latin America. In the case of Spain, 73% of the population uses WhatsApp communication platform (being the 89% of smart-phone users) by January of 2018.

1.1 Communications and offensive content

Although the use of social media and communication platform have improved the life quality and comfort of the citizens, it has also carried a lot of problems, specially the cyberbullying. This is generated most of the time due to the anonymity that these applications offer.

As [Coo] points, 32% of teens have been victims of some type of cyberbullying in 2007 and these stats have remained the same until 2016. In addition, 20 % of users were reported that they were affected by online rumors. Finally, as [ERK15] says, cyberbullying often occurs on Facebook or through text messages.

There are a lot of sources that claim that offensive content in social media and communication platform are very dangerous. For example, in [Sch] social media problems with public institutions (especially in election periods) are discussed. As it says, it's very complicated to handle the offensive content toward public institutions and when the opinion should be free or restricted. For example, some examples where the comments should be banned are exposed:

- When a poster is using vulgarity or insulting entire groups (sexism, racism, homophobia, etc.)
- When someone is threatening or implying violence
- When someone is launching specific and clearly slanderous attacks on an elected official
- When a person is directing vicious personal insults at other commenters

On the other side, private companies are concerned with this problem too. For example, Alphabet (with YouTube adds) had a big problem with major brands' content was found to be appearing next to videos promoting extremist views or hate speech as it appear in [Sol].

1.2 Tellfy Communication platform

Tellfy is an effective communication platform for smart-phones created by NexTReT Company. From 2017, this communication platform works in both Android and iOS mobile platforms as well as a web page for communication and analytic dashboard. At the start of 2020, Tellfy communication platform had almost 5000 registered users, around 300 different communities and over 1600 groups.

As well as in most of the communication platforms, Tellfy includes both group and individual chats. In addition, these chats are enclosed within communities, where a group of people can join and share different groups for each topic (as well as private chats),

The main strength of Tellfy compared to the rest of communication platforms of the market is the analytic service that it includes. This analytic service is oriented to the administrator of the communities to see the activity of the community. The analytic service is deployed in a web page and each administrator is able to see the information of all the communities that administrates. Information can be seen grouped in all communities, in a single community or in a single chat group (only the public ones).

Tellfy servers

To sustain all this platform stable and working properly as well as ensure the scalability of the architecture, a set of servers and data bases have been implemented with different sections for each purpose. In the Figure 1.1image below we can see the overview of the entire system.

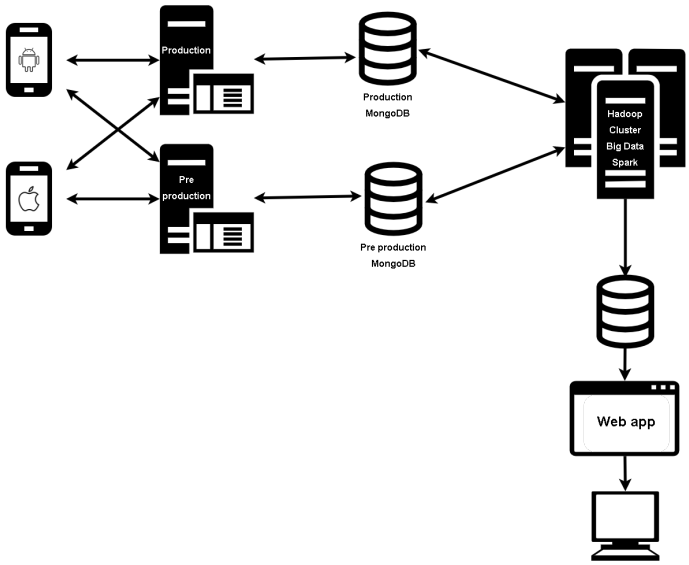

Figure 1.1: Tellfy platform structure blueprint

As it can be seen, the central part of the structure is a Big Data Hadoop cluster that uses Apache Spark to deal with data. This cluster is nowadays composed by 2 different machines, a master and a slave. The master is a machine with 16 cores and 32 GB of RAM memory. The slave is a machine with 2 cores and 64 GB of RAM memory. The main task of this Hadoop cluster is to save the generated data using distributed technology. This information is stored using Hadoop Distributed File System (HDFS) technology. Additionally, this cluster will have to save the information every 5 minutes of the data recorded in the MongoDB data bases of the smartphone applications.

In the other hand, this project has a 2 sets of server and MongoDB data bases for pre-production and production purposes. These sets will be connected in real time with all the

smartphones and will have to update, send and receive information from all the smartphones. As said previously, these two data bases (the one from pre-production and the other from production) will update every 5 minutes the information stored in the Hadoop cluster.

1.3 Privacy and Ethical aspects of content control

With the digitization of data in the last years and the increasing amount of personal data transferred and stored the privacy of people has been in doubt. In recent years few cases of appropriation and not ethical use of personal data have been happening [Fit]. For this reason people are very concerned about their privacy and want to keep their anonymity and their personal data safe in this aspect. In fact, as [Axe] points out, although 75% of consumer are determined to put their personal data in the internet the 88% of them requires transparency from the company as a requisite.

As in NexTReT and specially in the Tellfy Communication Platform, sensible information and message content of the people is stored, all the process of data gathering and manipulation has to be done carefully.

So as to manage the correct privacy of people, the GDPR (General Data Protection Regulation) was designed to harmonize data privacy laws across Europe to protect citizens' personal data and stand on a united front regarding every organization's approach to safeguarding the processing of personal data. It was approved on April 14, 2016 and has been enforced on May 25, 2018.

In this work, all the statements of GDPR are preserved, as all the data gathering is done from public chats with previous authorization by the users and none of the personal information is attached to any of the messages to preserve the anonymity. However, some aspects of data gathering and controlling the content of the users may be ethically doubtful and need to be explained.

First of all, classifying a content as inappropriate is not easy. Although we live every day in a more international world with the mixing of the cultures, the meaning of inappropriate may differ from one culture to another. This task of setting a threshold of what is inappropriate has been challenging and difficult specially to platforms with users from different cultures. For example, Facebook (with Instagram specially) has been dealing with conflicting topic such as banning nipples in the photos. In their internal guides, they have banned all the pictures (with some exceptions) that had a visible nipple. This have been criticised by a large amount of people. In the article of [Pau] this situation is analyzed and how the influencers

are forcing Facebook to change this norms.

In addition, as personal data does not belong to the user some times, this carries a lot of ethically doubtful actions from part of the data owner, typically the social media platforms. The data of the users, such as the interactions that they made are very valuable and nowadays these data can be sold in exchange of big amounts of money. This is a big problem because although we try to protect our personal data from being stolen, we can not do anything as users if these data are sold. This situation is explained in the article of [Gri] and the author questions who (govern institutions, industry and private markets) should be the one that must solve this problem.

1.4 Natural Language Processing

The detection of inappropriate text in a communication platform involved Natural Language Processing (NLP) techniques. For this reason, below NLP background is explained that later is going to be used.

N-grams

N-grams are contiguous sequences of N items from a given sample of text or speech. Those items can be phonemes, syllables, letters, words or base pairs according to the application and needs of the system and N is a integer number with the quantity of the items that are grouped.

Part-of-Speech tagging

Part-of-Speech (POS) tagging is a process in which a word is categorized lexically taking into account the meaning and the context of the word. This tagging is done to extract information of the structure of the sentence and language patterns.

Word Embedding

Word embedding are a type of word representation that allows words with similar meaning to have a similar representation. This representation of a word is given using a vector of a certain size where each element represent some information of the word. In these vectors, similar words will have as well most of the values equals or similar.

Chapter 2

Objectives

Tellfy communication platform is directed to both public and private formal institutions in which the content you can found has to be clean and without any inappropriate content.

This platform can be used in institutions such as schools, city councils or private companies to contact their customers, thus it is required that the public chats have not any offensive or inappropriate content in text, image, audio and video. As the main content in communication platforms usually come in text format, below we will focus on inappropriate text content.

As text is the main channel for communication in these kind of platforms, a lot of users take advantage of their anonymity to write in a inappropriate way and also send offensive content. The most common of inappropriate and offensive content are the following in social media and communication platforms by [SJ18]:

- Defamation
- Dehumanization
- Stereotyping
- Racism
- Profanity
- Negativity
- Capitalization
- Propaganda and Fake news

In addition, [WH16] points out that a tweet can be offensive if matches one of the following:

- Uses a sexist or racial slur
- Attacks a minority

- Seeks to silence a minority
- Criticizes a minority (without a well founded argument)
- Promotes, but does not directly use, hate speech or violent crime
- Criticizes a minority and uses a straw man argument
- Blatantly misrepresents truth or seeks to distort views on a minority with unfounded claims.
- Shows support of problematic hash tags
- Negatively stereotypes a minority
- Defend xenophobia or sexism

As it can be seen, these 2 papers agree on some of the topics that we found that are the most important and problematic in these kind of environment. For this present, the following topics and inappropriate text types have been defined as the ones to focus and detect them.

- Racist or sexist language
- Attacks to the minorities and stereotyping
- Negativity or attacking an individual

Although in these two previous articles different inappropriate content is distinguished, the big problem comes when identifying if one sentence is inappropriate or not. The different viewpoints between humans make this task really difficult as one sentence can be inappropriate for ones and not for other. In addition, the context and culture of each human, make this labelling more diffuse and difficult.

As the main objective of this work, a model able to classify any sentence in Spanish in appropriate or non-appropriate have to be created. In addition, a study of different type of techniques and models should be done to extract conclusions and start a new line of work and investigation in the NexTReT SL company.

Chapter 3

State of the Art

Inappropriate and offensive content detection in both text and image has been a inmense challenge and in the last few years huge advances have been achieved with the help of machine learning techniques. In the bibliography there are papers analyzing the situation and giving solutions to the problematic analyzed in chapter 2 with different approaches, mainly using machine learning algorithms.

In the following sections, different approaches and steps for inappropriate text content detection are reviewed:

3.1 Text processing

The text preprocessing is a necessary step in text classification tasks, and it is very important to choose both the tools that are going to be used as well as the performed steps.

Tools

There are many preprocessing tools available but in this case only 4 of the most used and suitable with the Tellfy environment are analyzed. As Python is a very used language in Data Science community and almost all the scripts in the Tellfy project are in this language, only tools with support for Python have been analyzed. In addition, some tools that use Spark are analyzed also taking into account their performance and features.

- SpaCy
- Spark NLP
- NLTK with Spark
- Apache OpenNLP

SpaCy is a open source Industrial-Strength Natural Language Processing tool that is widely used nowadays. This library provides a huge collection of various categories of NLP algo-

rithms. Moreover, SpaCy supports over 28 languages (including Spanish) and handles them very efficiently.

Spark NLP is an integrated tool with Spark and offers parallel computing of data. Using Apache Spark with power in memory clustering, it allows to perform data cleansing on large data sets. However, it doesn't include Spanish pre-trained model and requires a minimum version of Spark 2 to work (nowadays the Tellfy servers are running Spark 1.6).

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It has a possibility to include Spanish models from external collaborators as well as most common algorithms such as tokenizing, part-of-speech tagging and stemming. Although NLTK is not directed to large scale natural language processing, [Lab19] it's a collaborative project directed by John Snow that incorporates NLTK to Apache Spark so as to have an efficient and accurate NLP annotations for machine learning pipelines, that scales easily in a distributed environment.

Finally, Apache OpenNLP it's an open source toolkit for the processing of natural language text. OpenNLP supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, language detection and coreference resolution for English language. However, for Spanish language, only offers a model pre-trained with part-of-speech tagging. Although this last framework is designed to use with Java, some adaptations can be made so as to use it with Python.

[Rob] has compared the 2 tools that have a complete Spanish model (SpaCy and NLTK without Spark). In terms of performance, SpaCy has a better performance doing Word Tokenization and Part-of-Speech Tagging although NLTK offers better performance doing Sentence Tokenization.

Preprocessing steps

The first step to do with the original text data is to preprocess in steps, modifying the original text to a more comprehensible text for the machine. There are several studies and each of them applies different preprocessing steps. Below, some of the most used preprocessing steps are going to analyzed.

As [Dav] points out, some of the most used preprocessing steps for general purposes are the following:

- Convert all letters to lower or upper case
- Convert numbers into words or remove them
- Remove punctuation, accent marks and other diacritics
- Remove white spaces
- Expanding abbreviations
- Removing stop words, sparse terms, and particular words
- Text canonicalization

As these steps are directed for English language processing, some of them are not suitable for Spanish language. For example, in Spanish isn't very common to speak with abbreviations so the step of expanding abbreviations doesn't make sense.

[HGAQ18] explains the steps used for detecting aggressive tweets in Spanish language. These preprocessing steps are more similar to the ones that are going to be applied in this Master Thesis for two reasons. First, the language in both is Spanish and second, the goal of both studies is to detect inappropriate, offensive or aggressive language in text. In this study, the following steps are applied:

- Lowercase the text: Allows to improve the POS tagging process.
- Replace all digits by a single symbol: Avoids unnecessary confusions for the machine as the numbers do not carry semantic information.
- Remove @ from twitter mentions: Keeping the mention without the @ improves the POS tags features.
- Replace all picture links by a single symbol: As well as the previous case, avoid unnecessary confusions for the machine.
- Transform slangs: Replace slangs by their standard versions so as to have the most clear text as possible.

3.2 Text feature extraction

For the best possible classification of inappropriate sentences, feature (or characteristic) extraction of each sentence has to be made. In the following sections, the different kind of features extracted from sentences that are used in the State of the Art have been analyzed:

Language patterns

[HGAQ18] have performed a linguistic analysis of their training corpus in Spanish to identify the language patterns that can help distinguish aggresivity, detecting two types of them. Morphological structures and some recurrent lexical items, usually combined with some morphological category.

Character N-grams

[CT] works in text type categorization using N-gram Frequency. This N-grams are computed with characters series with sizes from 1 to 5. As they told, the primary advantage of this approach is that it is ideally suited for text coming from noisy sources such as email or OCR (Optical Character Recognition) systems.

In addition, [AGB18] also uses character N-grams with series from 1 to 3 for detecting hate speech and offensive language of Twitter. In the conclusions of the work, they point out that the optimal range for character n-grams are from 1 to 3, achieving up to 95.6% of accuracy.

Finally, in the work of [YLJ18] character N-grams are also tested. As they conclude, although character N-grams offer a very good performance in Neural Networks in general, they do not work so well in tasks like hate and abusive speech classification in Twitter.

Word N-grams

In addition to the use of character N-grams, [YLJ18] also tests the same models using Word N-grams, as well as hybrid version of both. After all the combinations results, they concluded that while character N-gram features are known to improve the accuracy of general purpose neural network models, word N-gram features perform better in hate and abusive speech classification.

Following the same trend, [SJ18] uses both character (sequences of 1, 3 and 5) and word (sequences of 1, 2 and 3) N-grams. As it can be seen in their results, the F1 score increases using both, compared to the same model without word n-grams (from 72,5% to 73,4%).

Part of Speech tagging

Part of Speech (PoS) tagging is largely used in the bibliography for text classification. For example, [CT]

In addition to the information that can be extracted from the Part of Speech, the combination of PoS with n-grams can extract language patterns that can be very valuable. In [LvR08] PoS n-grams are used to compute a term weight that represents how informative terms are in general. Their main argument is that PoS n-grams encode grammatical and structural information about language in a shallow way.

Especial characters

- Use of upper case
- Use of exclamation marks
- Use of any offensive word
- Use of many offensive words

Text Data Vectorization

For applying the different machine learning techniques the data features must be numerical or categorical. However, text or N-grams data are not either numerical nor categorical so there is a need to vectorize them. A vectorization is a process in which a numerical value is given to each of the words. As [Sha] analyzes, there are 2 main types of encoding that are used nowadays, Bag of Words (BoW in short) and Label Encoding.

The first one of them, is a basic mechanism used in natural language processing in which the order of the words in a sentence is not taken into account, but just if the word appears in the text or not. Initially, one corpus is analyzed and a column is created for each distinct word that is in the corpus. Then, for each sentence, if the word appears in the text, a value of *1* is given in that column or *0* otherwise.

The second one, generates a dictionary with all the words available in the corpus and assigns an index for each of the words. After that, a sentence is represented as a vector in which the elements are the indexes corresponding to each word in the dictionary. This method is widely used for classification problems.

Additionally, as [Pan] shows, there are many different implementations of these two encoding. For example, Scikit-Learn offers these different implementations based on Bag of Words:

- Count Vectorizer
- TFIDF Vectorizer

Count vectorizer is a simple implementation of a Bag of Words in which the number of occurrences of each word is used instead of a binary assignment (if the word appears or not in the sentence).

Term Frequency-Inverse Document Frequency (TFIDF) is a variation of Count Vectorizer that weights the importance of the word in the corpus or data set. TFIDF is data extracted from the combination of Term Frequency (TF) and Inverse Document Frequency (IDF). TF is defined as how frequently the word appear in the sentence. IDF is another concept which is used for finding out importance of the word. It is based on the fact that less frequent words are more informative and important.

In other hand, both Scikit-Learn and Keras (a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano) offers different text vectorization tools to do Label Encoding. In both cases, first a dictionary is given and an index is assigned for each word, later the text tokenized in words and is transformed to those indexes.

Word Embedding

In addition, another authors have developed different word embedding that offer great performance starting from Label Encoding. These are the most important and the most used:

- Word2Vec
- GloVe
- FastText

Word2Vec [Ten] is a complex word embedding model with two-layer neural network created by Google. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand.

GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embedding by aggregating global word-word co-occurrence matrix from a corpus. The resulting embedding show interesting linear substructures of the word in vector space [Chab]. Despite the different approach of both Word2Vec and GloVe, the results in both word embedding are very similar.

FastText is a library created by the Facebook Research Team for efficient learning of word representations and sentence classification. FastText (which is essentially an extension of Word2Vec model), treats each word as composed of character n-grams. So the vector for a word is made of the sum of these character n grams [Sub].

3.3 Text classification

Once the text is preprocessed and vectorized, machine learning techniques can be applied to classify the text. In this thesis, only binary classifications are going to be analyzed, as the final goal is to detect if a sentence is or not inappropriate.

[YLJ18] reviews most of the popular machine learning algorithms.

- Naïve Bayes (NB): Multinomial NB with additive smoothing constant 1.
- Logistic Regression (LR): Linear LR with L2 regularization constant 1 and limited-memory BFGS optimization.
- Support Vector Machine (SVM): Linear SVM with L2 regularization constant 1 and logistic loss function.
- Random Forests (RF): Averaging probabilistic predictions of 10 randomized decision trees.
- Gradient Boosted Trees (GBT): Tree boosting with learning rate 1 and logistic loss function.
- Convolutional Neural Networks (CNN): Word, Character and Hybrid level CNN models use cross entropy with softmax as loss function and Adam as optimizer.
- Recurrent Neural Networks (RNN): Bidirectional RNN as the baseline, implementing a GRU cell for each recurrent unit.

Using a data set of 70904 tweets classified into 4 labels (*normal*, *spam*, *hateful* and *abusive*), the previous machine learning algorithms were tested with Precision, Recall and F1 metrics and the results of the baseline models are shown in the Table 3.1.

Model	Precision	Recall	F1
NB (word)	0.747	0.767	0.741
NB (char)	0.752	0.751	0.744
LR (word)	0.786	0.802	0.780
LR (char)	0.788	0.804	0.783
SVM (word)	0.773	0.775	0.730
SVM (char)	0.778	0.781	0.740
RF (word)	0.767	0.781	0.745
RF (char)	0.765	0.789	0.760
GBT (word)	0.772	0.794	0.773
GBT (char)	0.770	0.791	0.772
CNN (word)	0.789	0.808	0.783
CNN (char)	0.768	0.787	0.747
CNN (hybrid)	0.790	0.807	0.781
RNN (word)	**0.804**	**0.815**	**0.804**
RNN (char)	0.367	0.605	0.457

Table 3.1: Comparison of different machine learning algorithms

As it can be seen in the Table 3.1, Neural Network algorithms are the ones with better performance (especially the RNN with words) with the exception of RNN with characters. Although RNN offers a really impressive performance with words, it has a really bad performance with chars. However, CNN offers a very good performance in all word, character and hybrid cases. Looking to the non-NN algorithms, although the performance is very similar, Logistic Regression offers a better performance compared to the rest of them.

Moreover, [AGB18] analyzes 3 of the previous algorithms (Naïve Bayes, Logistic Regression and Support Vector Machine) and their performance. In this paper, the classifier models are trained using n-gram, term frequency-inverse document frequency (TFIDF) as features and 3 different labels (*hateful*, *offensive* and *clean*). Although the performances in this case are very similar with different configurations of n-grams, the Naïve Bayes offers a slightly better performance than the rest. The best performance score was achieved with 1, 2 and 3 n-grams and L2 normalization (0.926 of accuracy), in addition a better performance was achieved smoothing Alpha value for the features (0.934 of accuracy).

As explained in the paper of [YLJ18], RNN are very effective machine learning algorithms and [GKPL18] proposes a Recurrent Neural Network that is composed of multiple Long-Short-Term-Memory (LSTM) based classifiers. As the paper states, the techniques that are used are not necessarily revolutionary in terms of the deep learning models used. However, it shows that the results are quite effective.

In addition, [GN18] uses again a bi-directional RNN as [YLJ18] and [GKPL18] do. But in this case using a semi-supervised training approach. Although the work of [Lee] is quite old, it also uses Pseudo-label techniques with Neural Networks.

3.4 Data augmentation

The most challenging aspect making a good inappropriate content classification for a language like Spanish is the lack of labelled data that is representative. In order to have as much as text quantity (without losing the quality of the text), two different approaches can be taken. The first way is generating and labelling manually the data, as it will be explained in the chapter 4. The other way is generating new data from previous original labelled data.

As [Cou] explains, these are the main techniques of data augmentation:

Textual noise injection

The injection of noise into a neural network can be considered as a form of data augmentation. It is possible to improve the robustness of a neural network by adding random noise to its inputs. This addition of noise generally contributes more to the robustness of learning than to the recognition of new forms in the data.

For example,

Nadie espera que Donald Trump gane el Nobel de la Paz.

to

Todos esperan que Donald Trump no gane el Nobel de la Paz.

Spelling errors injection

The idea is to simulate the most common spelling errors that human make while writing, specially when writing fast in informal context. This spelling injection algorithms are based commonly in a list of the most common mispellings in that language.

For example,

Aquí no lo tengo, pero voy a ver si lo tengo ahí.

to

Aquí no lo tengo, pero voy aver si lo tengo hay.

or

Aquí no lo tengo, pero voy haber si lo tengo ay.

Word replacing with thesaurus

Lexical or word replacement consists in changing the words by maintaining the same information of the sentence. In addition, including hyperonyms (a more general word) or hyponyms is also a possibility, although the second one is not recommended by [Cou].

For example,

Me voy a casa en un coche.

to a synonym

Me voy a casa en un automovil.

or to a hyperonym

Me voy a casa en un vehiculo.

or to a hyponym

Me voy a casa en un Fiat Multipla.

Chapter 4

Data

The data is the most important part for making prediction or classification systems. Any machine learning algorithm needs data in order to learn patterns from it. We can divide the ML algorithms in two types; supervised and unsupervised [Son].

The main difference between the two types is that supervised learning is done using a ground truth, or in other words, we have prior knowledge of what the output values for our samples should be. Therefore, the goal of supervised learning is to learn a function that, given a sample of data and desired outputs, best approximates the relationship between input and output observable in the data. Unsupervised learning, on the other hand, does not have labeled outputs, so its goal is to infer the natural structure present within a set of data points.

Although the languages have patterns and some information can be extracted using unsupervised machine learning, detailed and concise information as inappropriate language use have to be extracted using supervised machine learning.

As most of the community and chats in Tellfy use Spanish language (although in some of them is shared with Catalan with a minor use), the offensive content detection is going to be only in Spanish language initially. However, as the language structure in both languages is very similar, a future model with Catalan language could be generated easily because the automatic translations are almost perfect. For generating a Spanish corpus, the next sources have been used:

4.1 Corpus from other authors

Initially, the first step was to check the Corpus used by other authors in the State of the Art. Although a large amount of Corpus with offensive content labeled was found in English, none of them was found in Spanish. For this reason, English corpus was translated

to Spanish. This decision was taken because in some bibliography this step was done to do Data augmentation (as it can be seen in section 3.4) without losing or altering information.

For this purpose, the Google API for translation was used and a data set of 30000 sentences were translated to Spanish. This step was crucial because it helped a lot to have easily a Corpus with 30000 sentences with a proportion of a 25% of inappropriate sentences. However the translation was confusing especially in the cases of inappropriate sentences.

For example, the next sentence is offensive, beeing the meaning of *tosser* translated to *lanzador* that's not offensive, when it should be translated to *capullo* or *gilipollas*.

I don't want to get out with you, you're a tosser

Translated to Spanish using the Google API:

No quiero salir contigo, eres un lanzador

In addition, some of the labels where wrong and this reduces the overall performance of the system. For example, the following sentence was labeled as *no hate*:

And to all you hudeksem , you know I 'm pulling your dicks ...

Translated to Spanish using the Google API:

Y a todos ustedes, hudeksem, saben que estoy tirando de sus pollas ...

As it can be seen, using this translated data set wasn't the best option because the Corpus wasn't good enough.

4.2 Twitter texts

After a failed attempt of retrieving labeled Corpus, the next step was retrieving tweets (that are very similar to communication short messages) in Spanish. These tweets were found searching for controversial topics such as Spanish election with the hashtag *#28M*. Those tweets were labeled manually by the author separating them into *inappropriate* and *appropriate*. The input values for each of them have been as follows:

- Value of 0 for *appropriate* sentences
- Value of 1 for *inappropriate* sentences

This corpus was completed with a total amount of 1440 sentences distributed as follows:

- 612 *appropriate* sentences (42% of the total)
- 828 *inappropriate* sentences (58% of the total)

4.3 Tellfy texts

With the aim of having a larger corpus and more similar to the input data that will have to classify in the application, the conversations of public groups from the application Tellfy are used. The historical messages from 2018 have been labeled by the author manually, having a initial set of 5664 messages have been collected distributed as follows:

- 5281 *appropriate* sentences (93% of the total)
- 383 *inappropriate* sentences (7% of the total)

4.4 Manually generated texts

To complete the corpus of Twitter and Tellfy data, data from different sources have been used (conversation examples in Internet, simulated conversation between people...). These messages are conversations simulating controversial situations. This message set of 362 messages have been distributed as follows:

- 237 *appropriate* sentences (65% of the total)
- 125 *inappropriate* sentences (35% of the total)

4.5 Train, Validation and Test splits

After joining all the data sources, the following data set of 7466 sentences with the different label proportion has been achieved:

- 6130 *appropriate* sentences (82% of the total)
- 1336 *inappropriate* sentences (18% of the total)

This data set has been randomly split, maintaining the proportion of each type, in the following 3 splits:

- Train: For feeding the model for the fitting

- Validation: For comparison purposes between different models and the hyper parameters of these.
- Test: Only opened in the final part of the project that have been used to test the final performance of the final models

The train split covers 5919 of the sentences, the 80% of the total amount with the following proportion:

- 4865 *appropriate* sentences (82% of the total)
- 1054 *inappropriate* sentences (18% of the total)

The validation split has 738 of the sentences, the 10% of the total amount with the following proportion:

- 607 *appropriate* sentences (82% of the total)
- 131 *inappropriate* sentences (18% of the total)

Finally, the test split has 739 of the sentences, as the validation split the 10% of the total amount with a very similar proportion:

- 608 *appropriate* sentences (82% of the total)
- 131 *inappropriate* sentences (18% of the total)

Chapter 5

Experimental methods

5.1 Text processing

After analyzing the different text processing tools in the State of the Art (in section 3.1), prioritizing the performance of each of them the Spacy tool has been selected to perform text processing tasks. Spacy offers support in Spanish and handles with efficiency the tasks. As this inappropriate content detection has to be done as fast as possible, Spacy will be suitable for this purpose. For the text preprocessing, the following steps have been proposed:

- Convert all the text to lowercase
- Replace all the numbers (both in string or numeric format) by a symbol. As [AGB18] points out, this avoids unnecessary confusions for the machine.
- Replace all the mentions (in Twitter all the words starting with @) by a symbol for the same reason of the numbers.
- Remove punctuation
- Remove URLs for unnecessary confusions. As [AGB18] explains, text without any meaning or information can produce confusions for the machine.
- Remove stop words. As removing stop words for improving model performance isn't proved, both forms are going to be tested (with and without stop words).

5.2 Text feature extraction

Data features are a key component to extract valuable information. Feature extraction in text data nowadays is not clearly defined but some patterns are repeated in most of the works. In this case, following the work of [SJ18], the use of the following n-grams (for both characters and words) is going to be tested to compare the performance.

- Character 1-grams

- Character 3-grams
- Character 5-grams
- Word 1-grams
- Word 2-grams
- Word 3-grams

In addition, the same work uses metrics of especial characters for detecting offensive tweets. As it is explained in the work that this gives an additional boost in performance, the following especial cases metrics are going to be used extracted from the original sentences before preprocessing the text:

- The sentence uses more than a third part of text in upper case
- The sentence contains at least 3 exclamation marks
- The sentence contains at least one offensive word (inside a predefined list of 183 offensive words)
- The sentence contains at least three offensive words (inside a predefined list of 183 offensive words)

Finally, the use of Part of Speech tags of the text is also very common in some of the works of the State of the Art. For this reason, a POS-tag analysis is going to be performed to see the how it performs in addition to the other features. The POS-tag analysis will be performed using tri-grams of word POS-tags that have been extracted using the library of SpaCy.

Text data vectorization and Word Embedding

As explained in the section 3.2, the sentences need to be vectorized and different options have been analyzed in the State of the Art. The implementation of Count Vectorizer of Scikit-Learn is the basic model for word vectorization and for this reason one of the most used in natural language processing. Although it is widely used, the memory (and therefore computational) cost that requires is very high as the vector usually contains very few values greater than zeros compared to the total number of zeros. In addition, this vectorization method ignores the order of the words in the sentence so some information is lost in this process.

TFIDF Vectorizer continues having the same problem of generating very sparse arrays in

large corpus. However, TFIDF gives more information about the importance of some rare words found in the corpus compared to the simple count that simple Bag of Words methods does.

Finally, in contrast to all Bag of Words based encoders, Keras Vectorizer is a simple but very efficient implementation of Label Encoder. For this reason, is widely used nowadays and offers very good results. However, as it assigns a number to each word randomly, the similarities between words is lost. As in each feature the word that is assigned is different, it does not make sense to apply it in machine learning models like Random Forest or Logistic Regression nor in Feed Forward Neural Networks. However, this encoding is very used in Deep Neural networks like Convolutional or Recurrent Neural Networks.

To solve this problem of losing the information of the words and therefore the similarities between different words, word embedding techniques are going to be applied that are proven to be very efficient and techniques that offer very good performance. As Word2Vec and GloVe results are very similar, only Word2Vec will be analyzed and tested in this work. For this reason, the differences between Word2Vec and FastText will be shown.

FastText generates better word embedding for rare words (even if words are rare their character n-grams are still shared with other words, hence the embedding can still be good). What is more, FastText offers the possibility to represent out of vocabulary words as they are constructed from the n-grams instead of the words directly. Finally, the usage of character embedding (individual characters as opposed to n-grams) for downstream tasks have recently shown to boost the performance of those tasks compared to using Word2Vec or GloVe [WLT16].

Initially, Count (BoW type encoder) and Label encoders are going to be analyzed. In addition, if Count Vectorizer offers a good performance, the variation of TFIDF will be tested. Separately, the performance of the two different word embedding (Word2Vec and FastText) are going to be tested using Label Vectorizers.

For comparing the word embedding with the most similar format as possible, the Spanish Word Embedding offered by [JPRZ19] were planned to be used that are trained all with Spanish Billion Word Corpus and have the following format:

- Word2Vec: 1000653 vectors of 300 dimensions
- GloVe: 855380 vectors of 300 dimensions
- FastText: 855380 vectors of 300 dimensions

However, after checking if this corpus would be representative in our data it has been discarded. It has been seen that over a 20% of the words (most of them inappropriate words and colloquialism) are not represented in those word embedding. For this reason, a own

word embedding will be generated using corpus from Twitter.

To create the Word2Vec and FastText word embedding, the Gensim tool have been used training it with corpus extracted from Spanish Twitter. The selected dimension to represent each vector have been of 300, as most of the works and specially the data set of [JPRZ19] are also with this format. The number of vectors generated have been of 130861. Although it below the number of vectors of the previous data set, in this case the information that contains is more representative.

5.3 Metrics

The main goal of this project is to achieve a model that is able to detect with the best precision the possible inappropriate contents in the text. The main premise is to increase as much as possible the True Positive rate, maintaining as low as possible the False Positive rate. As different configurations with different model types are going to be analyzed, some metrics will be defined so as to evaluate properly each of the configurations. During the executions the following metrics are going to be calculated:

- Accuracy
- Recall
- Precision
- F1

Although the Accuracy metric is one of the most used, when the data set is not balanced it doesn't provide a meaningful information. On one hand, Recall is the fraction of the total amount of relevant instances that were actually retrieved. On the other hand, Precision is the fraction of relevant instances among the retrieved instances. The main advantage of Recall metric, is that it provides us with very useful information as we can know how many inappropriate sentences have been classified properly. However, it doesn't include information about the false positives as it does the Precision.

To resume both Recall and Precision metrics, F1 has been selected. F1 is the harmonic mean of Precision and Recall:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{5.1}$$

Using F1 as the main metric, we will consider both premises previously written, increasing

True Positives and decreasing False Positives. In addition, Recall and Precision metrics are also going to be taking into account with lower importance. Finally, Accuracy and execution times are going to be shown as extra information despite they won't be used for comparison purposes between different configurations.

To establish a baseline level for our model, 2 papers have been selected ([SJ18] and [YLJ18]) that have achieved the following results. The first one, arrives to a 88.9% of F1 score using a data set of 5000 sentences (a ratio of 1/3 for offensive sentences) using both feature selection and sentiment analysis. The second one, uses 4 different labels with a data set of 70 thousand sentences (60% of no-offensive sentences). After trying different features and models, it arrives to a F1 score of 81% using RNN.

5.4 Text classification

As there are a huge variety of text classification algorithms, the State of the Art has been analyzed and only the most used and the ones with the best performance are going to be implemented and used to analyze their performance. As it has been said in the section 3.3, the Convolutional and Recurrent Neural Networks are the ones with the best performance. In addition, Logistic Regressions are also going to be analyzed for comparison purposes between different types of algorithms. All the Neural Network models used in this section are extracted and adapted from [Tea19]. Also, they are going to be tested with a binary cross entropy loss and Adam optimizer in the compiler.

5.4.1 Feedforward Neural Networks

As an example of a Feedforward Neural Network, a 2 inner layer simple adaptation has been implemented. The model has the following layers:

- Dense layer with 64 units and ReLU activation
- Dense layer with 16 units and ReLU activation
- As output a dense layer with a single unit and Sigmoid activation

5.4.2 Convolutional Neural Networks

So as to test a Convolutional Neural Network, the following configuration has been implemented based on the article of [Jan]:

- Embedding layer with output dimension of 64

- Dropout of 0.2
- Conv1D layer with 250 filters, kernel size of 3 and valid padding
- GlobalMaxPooling1D layer
- Dense layer with 250 units
- Dropout layer of 0.2
- ReLU activation layer
- Dense layer with a single unit and sigmoid activation

5.4.3 Recurrent Neural Networks

To test the Recurrent Neural Network a model with a LSTM layer is going to be used. For the model with LSTM, the following configuration has been implemented, based on the article of [Li]:

- Embedding layer with output dimension of 128
- LSTM layer with 128 units, dropout of 0.2 and recurrent dropout of 0.2
- Dense layer with a single unit and sigmoid activation

5.4.4 Convolutional Recurrent Neural Networks

To test mode advanced and complete models, 2 different Convolutional Recurrent Neural Networks have been implemented.

CNN with LSTM layer

This model will be composed by the following layers, based on the article of [Bro]:

- Embedding layer with output dimension of 64
- Dropout layer of 0.25
- Conv1D layer with 64 filters, kernel size of 5, valid padding and ReLu activation
- MaxPooling1D layer with a size of 4
- LSTM layer with 70 units
- Dense layer with a single unit with a sigmoid activation

CNN with GRU layer

This model will be composed by the following layers, based on the article of [S.]:

- Embedding layer with output dimension of 64
- Conv1D layer with 32 filters, kernel size of 3, same padding and ReLu activation
- MaxPooling1D layer with a size of 3
- Dropout layer of 0.3
- Conv1D layer with 32 filters, kernel size of 3, same padding and ReLu activation
- MaxPooling1D layer with a size of 3
- Dropout layer of 0.4
- GRU layer with 50 units
- Dropout layer of 0.25
- Flatten layer
- Dense layer with 128 outputs and a ReLu activation
- Dropout layer of 0.45
- Dense layer with a single unit with a sigmoid activation

5.4.5 Logistic Regression model

To develop a Logistic Regression model, the implementation of the Scikit-Learn library have been selected because of its performance, robustness and the easiness to use the library. In this case, after seeing the different works in the State of the Art, the following combination of hyper parameter have been selected (the ones used in [YLJ18]), using the rest with their default values:

- Penalty of l2
- Tolerance of 0.0001
- Lbfgs as an algorithm to use in the optimization problem

5.4.6 Random Forest model

As well as with the Logistic Regression model, the implementation of Scikit-Learn library have been used. Again, the works of the State of the Art have been analyzed to select the

best hyper parameters. Also following the work of [YLJ18], the following hyper parameters have been selected and left the rest with their default value:

- 100 number of estimators (10 times more than the 10 used in the work, to maximize the generalization)
- 8 features for each estimator

Chapter 6

Results

The main objective of the present thesis is to find the best model to detect inappropriate content, by comparing multiple models and NLP processing techniques. For making some organized combinations, different phases have been designed using a set of locked and free variables in each of them for testing only few variables in each. The following phases have been realized and are explained through this chapter:

- Phase 1: Model testing with 4 different feature combination and 2 different encoding
- Phase 2: Test Neural Network models with Word Embedding
- Phase 3: Use alternatives to the base encoding (TF-IDF)
- Phase 4: Test the use of class weight for supplying the lack of balanced data set
- Phase 5: Use different length of data set and check results with different models
- Phase 6: Model testing with different N-gram combinations as features
- Phase 7: Create an ensemble model and compare results using test set

In the first 6 phases, all the results that are showed are tested using the validation set that is explained in the chapter 4. In addition, during the phase 7, this validation data is used for training and the test that is explained in the same section is used to see the performance of the models.

6.1 Phase 1: Model, feature and encoding testing

The aim of this first phase is to test the performance of the different models defined in section 5.4 with different features and encoding. After this phase, the best combinations between models and encoding are going to be selected for further testing with different configurations. The different encoding algorithms that have been used are Scikit-Learn Count Vectorizer and Label encoding. Finally, the feature combinations that are going to be

used in this phase are the following (in posterior phases, the feature combinations are going to be tested in deep):

- Word: Only preprocessed words
- Word & Metrics: Only preprocessed words and metrics of special characters (explained in the section 5.2)
- Word, Metrics & N-gram: In addition to the previous features, character tri-grams are added.
- Word, Metrics, N-gram & POS: In addition to the previous features, Part of Speech tag of each word is added as feature.

Results with Bag of Words encoding

The first part of this phase has been executing all the models with Count Vectorizer of Scikit-Learn. In the figure below we can see the F1 score resultant of the executions, removing the CNN and LSTM models from the figure because the results have been equal to 0 (more detailed and complete results in the tables of appendix section A). As it can be seen in the Figure 6.1, FNN type model offers very good results using Count Vectorizer, having a F1 score of 0.7 with the best feature combination.

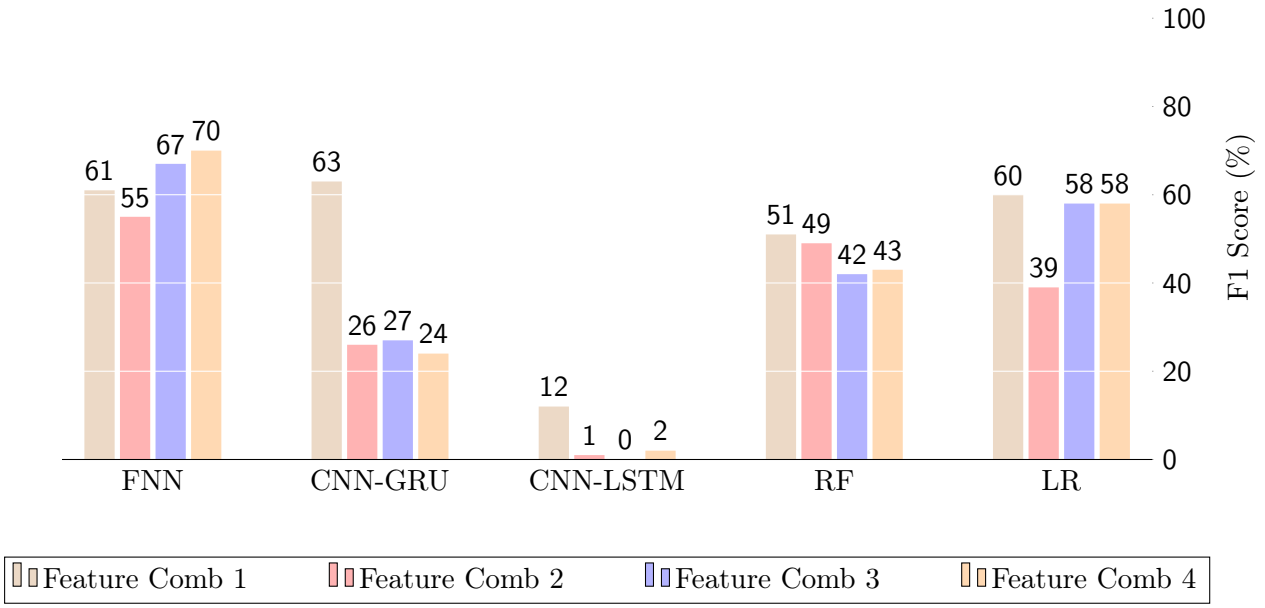

Figure 6.1: F1 score with different model and feature combinations using BoW encoding

In addition, it is surprising how CNN and LSTM type models have not been able to learn as their F1 score is 0 in all the feature combinations. Moreover, although CNN-GRU and

CNN-LSTM type models have achieved to learn, their performance is very low with the exception of the CNN-GRU with the first feature combination (only tokenized words as features).

Finally, even though the performance of Random Forest and Logistic Regression are not the best (specially in the case of RF), their performance is adequate compared to the Deep Learning models. Comparing both models, LR offers better performance in almost all the cases, specially with the first feature combination, arriving to a F1 score of 0.6. Taking into account the performances explained above and with the aim to select one model of each type (basic Neural Network, Deep Neural Network and Not Neural Network models), the selected combinations of models with features that are going to be tested more deeply are the following:

- FNN model with feature combination four.
- CNN-GRU model with feature combination one.
- LR model with feature combination one.

Results with Label encoding

In the second part of this phase, only the Deep Neural Network models have been used to test with Label encoding. FNN, RF and LR models have been discarded as these kind of models are not adequate using Label type encoding since each of the features will be assigned to different information each instance and therefore it will be impossible for them to learn any patterns. After executing all the feature combinations using Label encoding the extracted results are more similar between the different models despite the Bag of Words type encoders.

As it can be seen in the figure below (additionally, more detailed information can be found in section A), CNN and LSTM type models are the ones with the best F1 score, specially with the fourth feature combination. Although the performance in those models is slightly better than in the others, the computational cost (and therefore the time cost) is much higher than in the CNN-GRU and CNN-LSTM models. For this reason, these models (specially CNN-LSTM with the third feature combination) have to be considered as the performance and computational cost ratio is much better than the first two models.

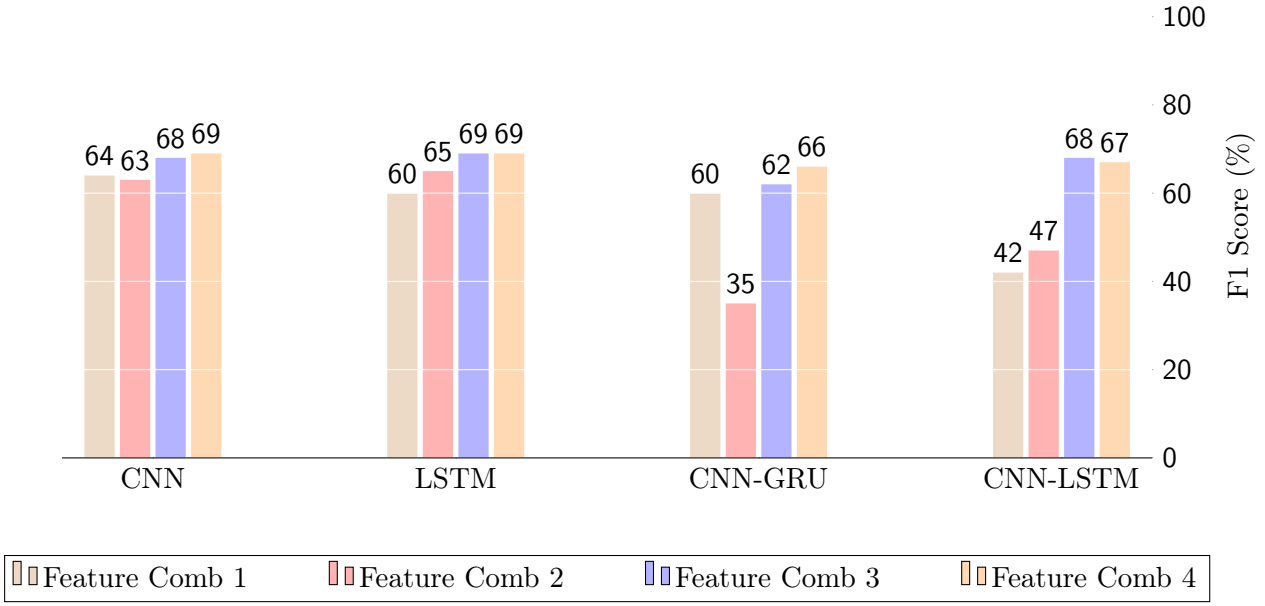

Figure 6.2: F1 score with different model and feature combinations using Label encoding

Additionally, as it can be seen in the tables Table A.9 and Table A.10, in the first 2 models (CNN and LSTM) the recall is much higher than in the other models with the fourth feature combination. In contrast, CNN-GRU and CNN-LSTM show very good (even better than the CNN and LSTM models) precision results using the fourth feature combination.

As all these four models are going to be tested using word embedding, the selection of the best model is going to be after that phase as the configuration of the models is the same in both cases.

6.2 Phase 2: Word embedding testing

As Bag of Words and Label based encoders are very simple and does not have the ability to represent each words meaning, in this phase two different Word Embedding are going to be tested, Word2Vec and FastText.

These Word Embedding are going to be tested only with Deep Neural Network models (CNN, LSTM, CNN-GRU and CNN-LSTM) as there are the ones that are able to extract benefit from having the words represented in a vector space.

In the figure below we can see the F1 score of the four models using Word2Vec and FastText

(more detailed information with different metrics in Appendix B). Opposite to the previous result phase with the Label encoding, using FastText as embedding, LSTM offers better results than the rest. It is clearly visible that Word2Vec offers a very poor performance compared to both the basic Label encoding and the FastText embedding. The reason of having so low values with Word2Vec could be due to the used vocabulary in the corpus as much of the words are badly written and they are badly represented in the vector space.

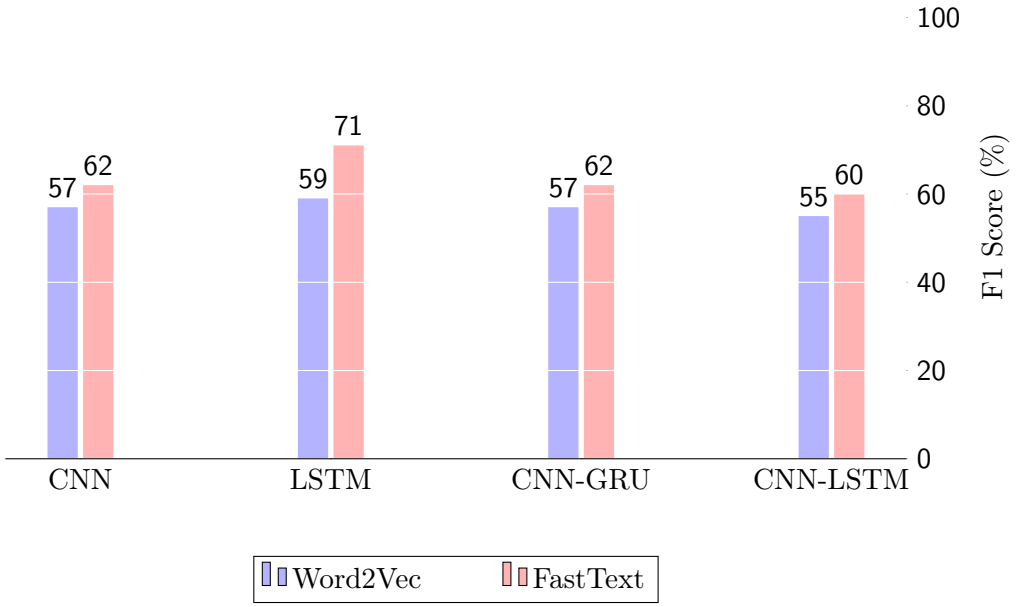

Figure 6.3: F1 score with different model and word embedding

In contrary, as FastText uses N-grams to allocate the terms in the vector space, the results are significantly better in the case of the LSTM model. For this reason, the model of LSTM with FastText as Word Embedding has been selected for further tests.

6.3 Phase 3: Bag of Words encoding types testing

After seeing the performance of the different models with different encoding and features, the selected models with Count encoding are going to be tested using the alternative Bag of Words algorithm, Term Frequency-âĂŞInverse Document Frequency, to see if the performance could be increased using this encoding. As it has been said previously, the following tests are with the following features:

- FNN: With the fourth feature combination.
- CNN-GRU: With the first feature combination.

- LR: With the first feature combination.

Although the use of TF-IDF is widely spread and a better performance compared to the Count Vectorizer is shown in most of the cases, in this case a considerable decrease can be seen in the figure below (more detailed metrics in Appendix C). Although in the case of the FNN the decrease is not considerable, in the case of LR is huge (dropping from 0.6 to 0.43) and in the case of CNN-GRU the model does not start to learn with TF-IDF. One of the possible reasons of having the CNN-GRU model incapable of learning is because TF-IDF gives lower importance to common words in the corpus and in our corpus offensive or inappropriate words are common. For this reason, the use of TF-IDF has been discarded for the following tests.

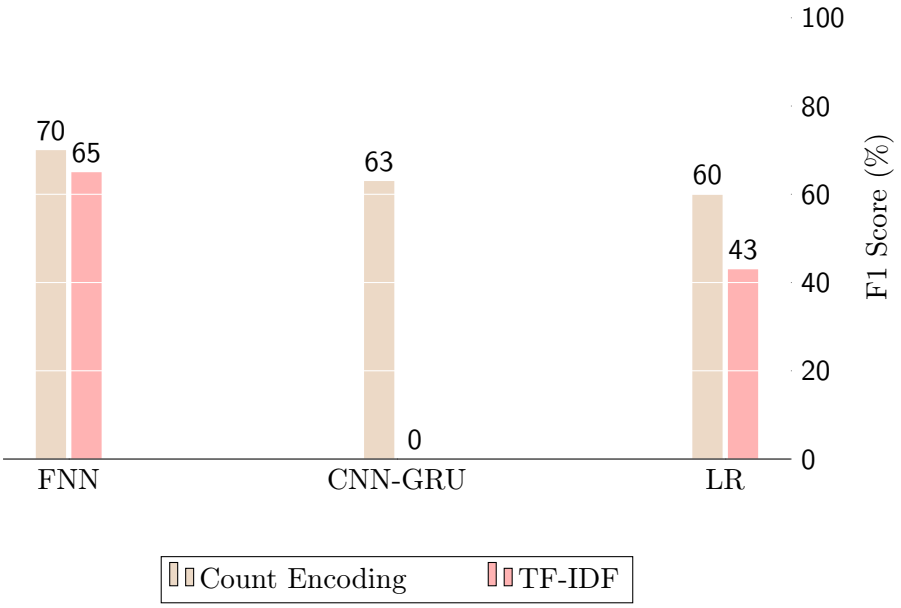

Figure 6.4: F1 score with different model and BoW encoding algorithm

As it can be seen, in all the cases the Count Vectorizer offers better results than TFIDF Vectorizer, so in future tests these models are going to be tested with Count Vectorizer.

6.4 Phase 4: Label weight testing

After seeing the performance of the different models with different encoding and features, the selected for further tests will be the following:

- FNN with all features and Bag of Words (Count encoding)
- CNN-GRU with word features and Bag of Words (Count encoding)
- LR with word features and Bag of Words (Count encoding)

- LSTM with FastText

In this fourth phase, all of these models are going to be tested using non-uniform label weights. As the data set is clearly unbalanced (inappropriate sentences are slightly more than a quarter of the data set), is expected that the models will learn better the inappropriate sentences. A 4 to 1 ratio have been selected (inversely proportional to the data set mix), giving a higher value to the inappropriate sentences.

As it can be seen in the figure below (more detailed metrics in the tables of the Appendix D), the use of weighted labels have improved the previous performances in all the models. It is visible that the one with the biggest improvement using weighted labels have been the Logistic Regression as well as the CNN-GRU with a lower increase. For this reason, and because none of the models have had a decrease in performance, the weighted ratio of 4 to 1 is going to be used in the rest of the phases.

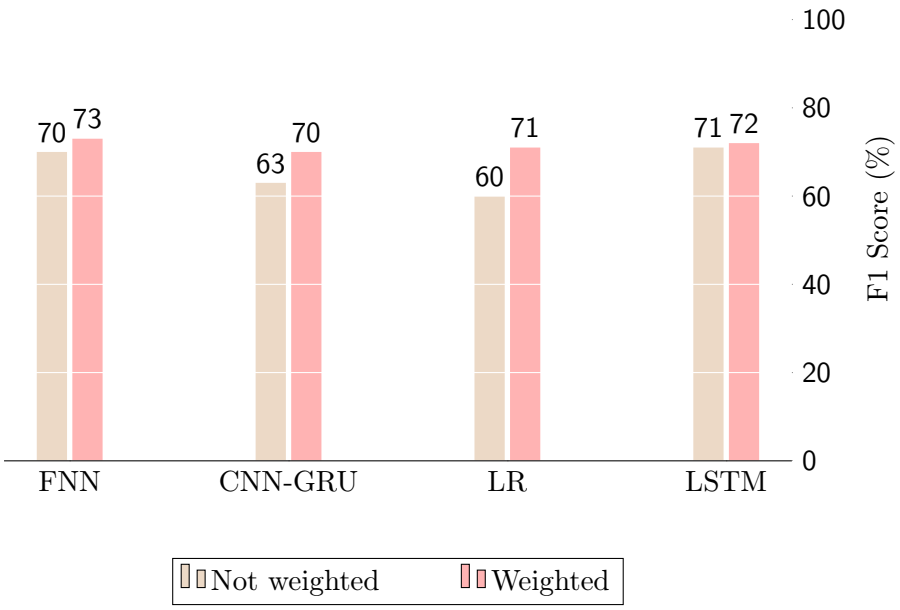

Figure 6.5: F1 score with different model and label weights

6.5 Phase 5: Different data set size testing

While reading the works of the State of the Art, one of the patterns that have been seen are the size of the training data sets. In almost all the cases, the training data sets had a minimum of 30 thousand sentences to a maximum of a hundred of thousand of sentences in some of them. For this reason, the training data set of less than 10 thousand sentences of this work seemed to be too small. To check if a bigger train data set was needed, the

four models have been tested using smaller data set samples to see the improvements while growing the data set. In the figure below, we can see the results of this experiment.

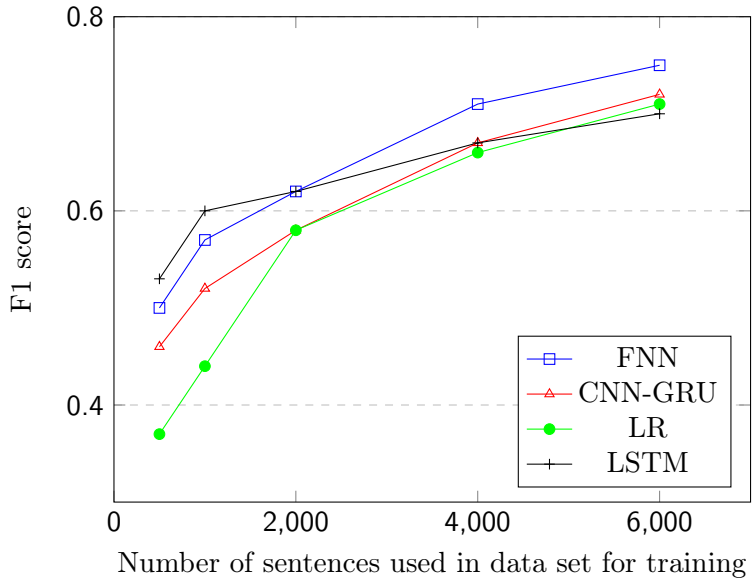

Figure 6.6: F1 score with different model and data set sizes

As it can be seen clearly, the performance increases considerably using more training samples. However, this phenomenon is slightly different in each of the models. For example, the LSTM model with the FastText word embedding, is the one with the best results using smaller data set, but the improvement adding more training data set is not as big as with the rest of the models.

In the opposite side, the Logistic Regression model offers a very bad performance using a small data set, but rapidly recovers the performance, having almost the same performance compared to the rest of the models.

Finally, FNN and CNN-GRU models show a very similar behaviour as they grow with a very similar shape, although FNN model offers a slightly better performance in all the cases.

In overall, we can see that most of the models increase considerably their performance from a thousand to a two thousand sentences data set and later the performance increase is not that high. However, in the last step (from 4 thousand to 6 thousand), all the models keep increasing their performance, so an additional test would be interesting using a bigger data set of 8 or 10 thousand sentences.

6.6 Phase 6: Different feature types testing

With the aim to improve the models performance, a feature selection of all the possible combinations have been done. The used features are the following:

- C1: Character 1-grams
- C3: Character 3-grams
- C5: Character 5-grams
- W1: Word 1-grams
- W2: Word 2-grams
- W3: Word 3-grams
- Special: Combination of the special features (UP, !!, O? & O+)

6.6.1 Part 1

As well as with the data set size, the State of the Art has shown that a huge variety of features are used for text classification and each work achieves better performance with different feature selections. For this reason, a forward feature selection will be done to achieve the best performance possible in the models of FNN, CNN-GRU and LR (LSTM will not be included as it works with FastText as embedding). As traditional forward feature selections, in the first part each of the features have been tested individually achieving the following results.

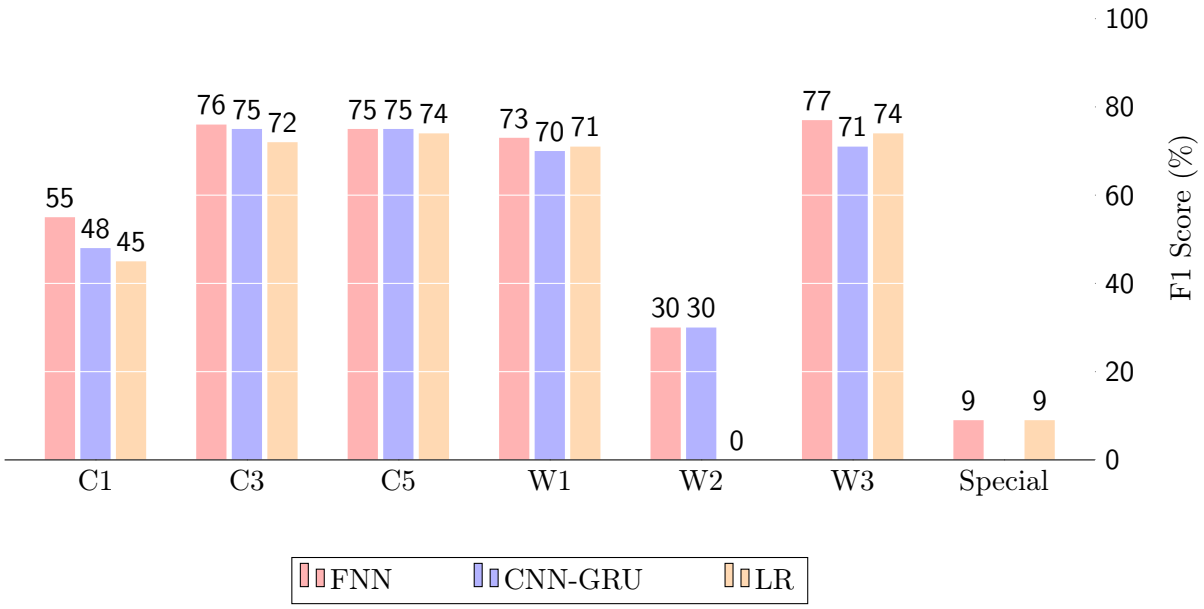

Figure 6.7: F1 Score of different model and feature selection

As it can be seen, 2 patterns are clearly visible. The first one, there are 4 features that achieve good performance results and are very similar between them (C3, C5, W1 and W3). The second one, all the features are performing with similar behaviours in the three models. To the next part of the forward feature selection, between C3 and C5 features we have selected the second one because the performance is better in overall in all the models. In addition, for further tests, only FNN and LR models are going to be used due to the computational cost that requires the CNN-GRU model. As the behaviour pattern of the three models is similar, it is not expected to have big changes from one model to other.

6.6.2 Part 2

In this second part of the forward feature selection, C5 features have been combined with the rest of the features, achieving the following F1 score results:

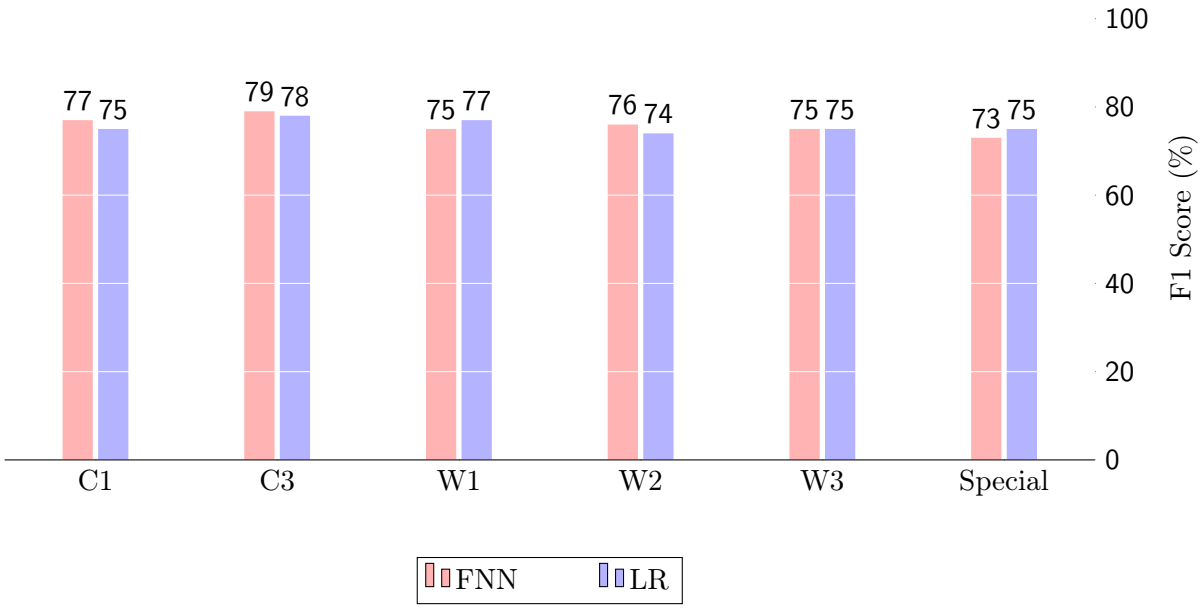

Figure 6.8: F1 score with different model and combinations of features with C5

Although the changes between the different feature combinations is not very big, there are some features that have better performance, for example, character based features. As character 3-gram feature combination is the one with the best performance and having an increase in the performance (increasing 4 points from using only 5-gram feature), the following part will have combinations of features using both character 3 and 5 grams.

6.6.3 Part 3

In the third part of the forward feature selection, C3 and C5 features have been combined with the rest of the features, achieving the following F1 score results.

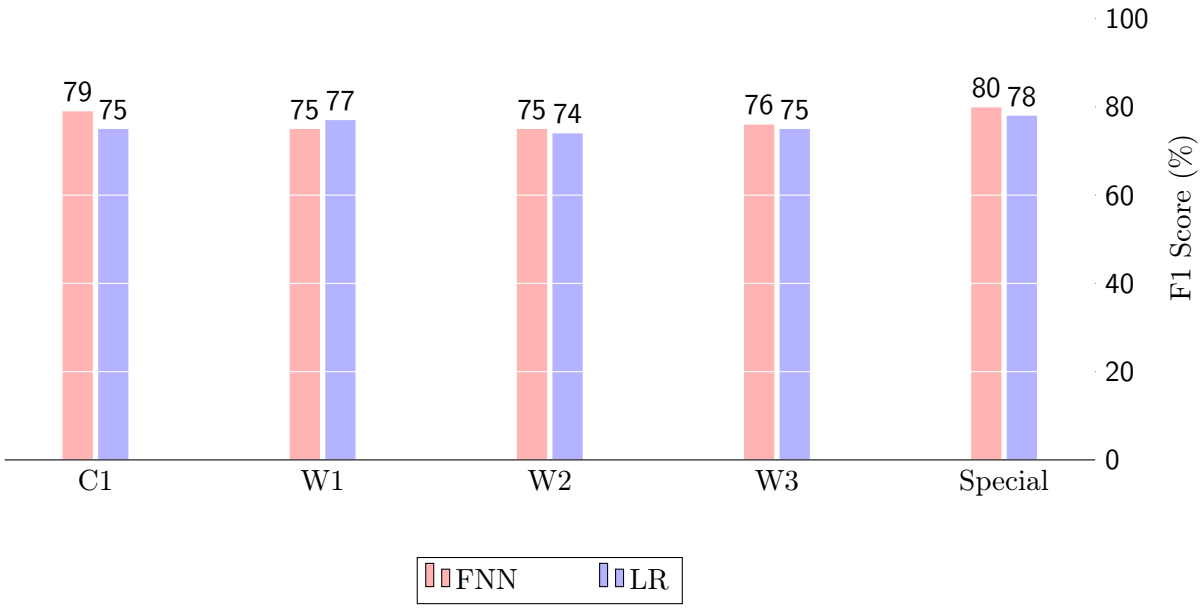

Figure 6.9: F1 score with different model and combinations of features with C3 and C5

As it can be seen in the figure above, word based features offer worst performance than only using C3 and C5 as features. In addition, special features improve slightly the performance of the C3 and C5 feature combinations. However, this increase is very low so the forward feature selection will be stopped here and the best model will be generated using this combination of C3, C5 and special features.

6.7 Phase 7: Testing ensemble models with test set

During the previous phases, 4 different models with the best feature and configuration combinations for them have been found. As these models are different between them so they are the results of the predictions (where some of them have greater precision rate while others have greater recall rate). For this reason, in this last phase, two different model ensembles are going to be tested in which the four models will be included.

The model ensembles will be created using for the first a decision tree and for the second a simple feed forward network (the same layer configuration as in the previous tests). These two model ensembles will be trained using the predictions of the validation set as features and will be tested using the test set that has not been used for training any of the models.

As it can be seen in the Figure 6.10, in addition to the decision tree and neural network ensembles, the rest of the four models also have been tested with the test set so as to see

a fair comparison in performance. In this figure we are able to see the F1 score of each of them using the test set. As expected, the results with the test and validation sets in the first four models are quite similar as both sets are extracted from the same original data set and therefore very similar.

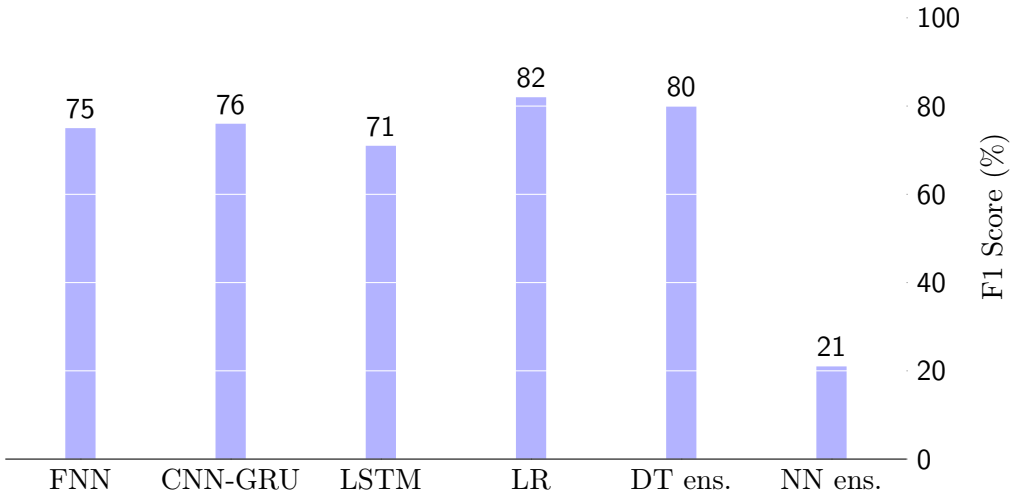

Figure 6.10: F1 Score of different models using test set

In the Figure 6.10 we can also see the different performance of both model ensembles. The decision tree shows a very good performance achieving a better performance than the rest of the neural network models, only being below the logistic regression model. In the other side, the neural network ensemble does not achieve a good performance achieving only 0.21 of F1 score. Comparing the results of all the models, it is clear that the logistic regression offers the best performance.

Finally, the rules learned by the decision tree ensemble have been extracted to see how offers that performance. As expected, the first node in the tree is the logistic regression feature (that is the one with the best performance) and although the rest of the features can change slightly the results, this feature will the most important one. The combination of the other three features will be used to make three exceptions. Although this exceptions should be used to improve the performance of the model, they do not. This behaviour can be due to the over-fitting to the validation set.

Chapter 7

Visualization

Apart from making the model and having it working with the best possible performance, a visualization of this have been done. The Tellfy Communication Platform has got a analytic web page in which user analytic can be seen. In this web page, a section to visualize inappropriate content have been developed using HTML, CSS and Bootstrap.

In the image below (Figure 7.1) we can see how the Android application looks like with an example of a conversation in a public chat. In this conversation, inappropriate content have been used to show how the model detects and visualizes it.

Figure 7.1: Screen shoot Android

Once the message has been sent, in the analytic web page we can see the detected inap-

propriate messages of the different communities and chats. The messages can be filtered by the last day, week or month. In the figure below (Figure 7.2) we can see how all the inappropriate messages have been detected with the corresponding confidence for each one.

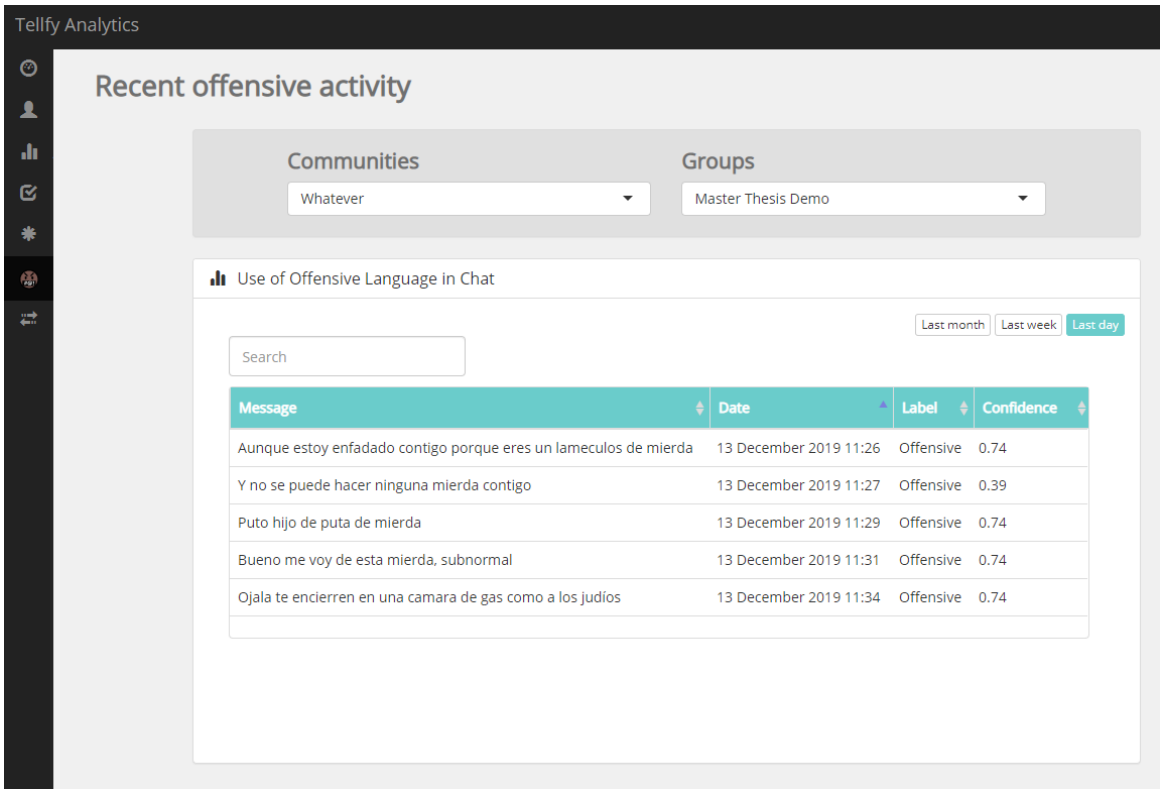

Figure 7.2: Screen shot web

Once the visualization of the images is done, a user is able to transform the predicted message in the web page (as in the Figure 7.3). Once a message classification is changed, it will be saved as wrong predicted in the data base with the correct label and will be available to use as training set in a future with this user feedback.

Figure 7.3: Screen shot web with changing prediction

Chapter 8

Conclusions

Nowadays the use of machine learning and natural language processing is starting to grow but there are not any general guides to how this technology performs in different context. As it have been seen during this work, some technologies work better in some circumstances, therefore the results found in the present study can not be extrapolated to some other circumstances. For this reason, the conclusions that are pointed out below are specific of this problematic.

One of the biggest problems that have been found during this project, it has been the lack of labelled public data sets in Spanish for inappropriate text classification. Although there are many big data sets in English for this purpose, the translations that have been tested (using Google Cloud platform) are not good enough and some of the inappropriate contents are lost in the translations. For this reason, a new data set has been created and labelled manually in the NexTReT SL company.

During the first phase of the testing, it has been seen that the text encoding type that is used is very critical and can change enormously the performance of the model. As each of the models works with a very different behaviour, it has been seen that Deep Neural Network models perform better using label type encoder, against the other models that perform better with Bag of Words type encoders. It must be said that the label encoders are much more efficient and therefore is preferable to use them when the performance difference is not huge.

Regarding the word embedding techniques, it has been seen that the benefits of using them is only visible when using LSTM type model. In the rest of the cases, a simple label encoding without any word embedding offers a better performance. One of the main reasons of this result may be due to the lack of corpus used in the generation of the word embedding space. As the corpus that it has been used is closely related and similar to the environment that the model will be placed, the short amount of text is not enough to generate a proper word embedding space with meaningful connections. To solve this problem and therefore increase

the performance of the model, a deeper scrapping on the net should be done to extract more corpus and generate a more precise word embedding

As said previously, the lack of a large data set has been a difficulty and therefore trying to maintain a balanced data set with a enough instances of inappropriate content have been impossible due to the lack of time. For this reason label weighting during the model creation has been used achieving impressive results in all the models. This weighting allows all the models to learn faster and also achieving a better performance.

When it comes to the training set length, during the different tests that have been done using different length sets it has been seen that the performance varies and has a clear correlation with the set size. However, not in all the models affects equally the size of the data sets. For example, LR is the one that shows a worst performance with less data although in the State of the Art the non-NN models are the ones with the best performance with small data sets. In the opposite situation we can find the LSTM model with the word embedding, that almost does not show any increase when using more instances. During this test, it has been seen that that the increase rate in the models does not decrease much with 6000 instances, for this reason more tests should be done with more instances and see the change in the performance in a future.

During the forward feature selection process, we have been able to see which of the used features were the most relevant and critical. As it was seen in the works of the State of the Art, character based N-grams were very effective. As well, as seen in some of the works, special metrics of the sentences offer some increase in the performance that is very useful in this context where the use of upper case and exclamations show aggressiveness.

Finally, in the phase seven of the testing, we have been seen that the model ensembles do not offer an increase in the performance compared to the best of the models (Logistic Regression model). The most probable reason of having this behaviour is because some of the predictions can not be done properly by any of the models. In addition, after all the tests, the most simple model and the one with the lower computational cost is the one with the best performance. This could be because the more complex models are not fine tuned and need a larger data set for being able to learn the patterns.

For future works, the failures that have been made predicting the validation and the test set must be checked and try to extract patterns. Knowing where the model fails, some modifications in the models can be done and increase the performance of them. In addition, although all the comparisons have been done using the metrics on the same data sets, some

models with some configurations may behave different in each split of data set. For this reason, all this study should be done using cross validation. The use of cross validation has been discarded from the start due to the lack of time and computational cost but it should be considered for further studies.

To conclude this work, it has to be pointed out that one of the performance levels established as baseline for our model has been surpassed by one point [YLJ18]. However, the results of almost 89% of F1 score of [SJ18] has not been able to reproduce. Nevertheless, all these results are using different data sets in different languages so the results should not be compared directly.

References

[AGB18] Shrikant Kendrez Aditya Gaydhani, Vikrant Domay and Laxmi Bhagwatx. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach, 2018.

[Axe] Axel. How concerned are consumers really when it comes to data privacy? *Medium*.

[Bro] Jason Brownlee. Sequence classification with lstm recurrent neural networks in python with keras. *Machine Learning Mastery*.

[Buc] Birgit Bucher. Whatsapp, wechat and facebook messenger apps | global messenger usage, penetration and statistics. *Messenger People*.

[Chaa] Dave Chaffey. Global social media research summary 2019. *Smart Insights*.

[Chab] Japneet Singh Chawla. What is glove? *Medium | Data Science*.

[Coo] Sam Cook. Cyberbullying facts and statistics for 2016-2018. *Comparitech*.

[Cou] Claude Coulombe. *Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs*. PhD thesis, Université TÉLUQ.

[CT] William B. Cavnar and M. Trenkle. N-gram-based text categorization.

[Dav] Olga Davydova. Text preprocessing in python: Steps, tools, and examples. *Medium | Data Science*.

[ERK15] Harmony Rhoades Hailey Winetrobe Jeremy Goldbach Aaron Plant Jorge Montoya Eric Rice, Robin Petering and Timothy Kordic. Cyberbullying perpetration and victimization among middle-school students. In *American Journal of Public Health*, pages 66–72, 2015.

[Fit] Sarah Fitzpatrick. Facebook to send cambridge analytica data-use notices to 87 million users monday. *NBC News*.

[GKPL18] Heri Ramampiaro Georgios K. Pitsilis and Helge Langseth. Detecting offensive language in tweets using deep learning, 2018.

[GN18] Isuru Gunasekara and Isar Nejadgholi. A review of standard text classification practices for multi-label toxicity identification of online content. In *Proceedings of the Second Workshop on Abusive Language Online (ALW2)*, pages 21–25, 2018.

[Gri] Rick Grinnell. The ethical use of data. *CSO United States*.

[HGAQ18] Gerardo Sierra Octavio Sánchez Helena Gómez-Adorno, Gemma Bel-Enguix and Daniela Quezada. A machine learning approach for detecting aggressive tweets in spanish, 2018.

[Jan] Nikolai Janakiev. Practical text classification with python and keras. *Real Python*.

[JPRZ19] Jose Cannete Jorge Perez Rojas and Zuik. Spanish word embeddings. https://github.com/dccuchile/spanish-word-embeddings, 2019.

[Lab19] John Snow Labs. Spark nlp. https://github.com/johnsnowlabs/spark-nlp, 2019.

[Lee] Dong-Hyun Lee. Pseudo-label : The simple and ecient semi-supervised learning method for deep neural networks.

[Li] Susan Li. Multi-class text classification with lstm. *Medium / Data Science*.

[LvR08] Christina Lioma and Keith van Rijsbergen. Part of speech n-grams and information retrieval. In *Revue française de linguitique appliquÃľe*, pages 9–22, 2008.

[Pan] Paritosh Pantola. Natural language processing: Text data vectorization. *Medium / Data Science*.

[Pau] Delia Paunescu. Inside instagramâĂŹs nudity ban. *Voc - Recode*.

[Rob] Robert. Nltk vs. spacy: Natural language processing in python. *The Data Incubator*.

[S.] Omayma S. Intro to text classification with keras (part 3 - cnn and rnn layers). *Once Upon Data*.

[Sch] Mike Schlossberg. When things turn nasty on social media. *Govering*.

[Sha] Raheel Shaikh. Choosing the right encoding method-label vs onehot encoder. *Towards Data Science*.

[SJ18] Tom De Smedt and Sylvia Jaki. Challenges of automatically detecting offensive language online, 2018.

[Sol] Olivia Solon. Google's bad week: Youtube loses millions as advertising row reaches us. *The Guardian*.

[Son] Devin Soni. Supervised vs. unsupervised learning. *Towards Data Science*.

[Sub] Nishan Subedi. Fasttext: Under the hood. *Medium / Data Science*.

[Tea19] Keras Team. Deep learning for humans. https://github.com/keras-team/keras, 2019.

[Ten] Tensorflow. Word embeddings.

[WH16] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of NAACL-HLT 2016*, pages 88–93, 2016.

[WLT16] LuÃŋs Marujo RamÃşn Fernandez Astudillo Silvio Amir Chris Dyer Alan W Black Wang Ling, Tiago LuÃŋs and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation, 2016.

[YLJ18] Seunghyun Yoon Younghun Lee and Kyomin Jung. Comparative studies of detecting abusive language on twitter. In *Proceedings of the Second Workshop on Abusive Language Online (ALW2)*, pages 101–106, 2018.

Appendix A

Appendix: Result tables of phase 1

Results with Bag of Words encoding

Loss

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	0.51	0.45	0.34	0.29
CNN	0.51	0.48	0.48	0.48
LSTM	0.49	0.48	0.55	0.48
CNN-GRU	0.54	0.52	0.52	0.55
CNN-LSTM	0.47	0.48	0.48	0.48

Table A.1: Loss with different model and feature combinations using BoW encoding

Accuracy

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	0.87	0.86	0.88	0.9
CNN	0.79	0.79	0.79	0.79
LSTM	0.79	0.79	0.79	0.79
CNN-GRU	0.88	0.81	0.81	0.8
CNN-LSTM	0.81	0.79	0.79	0.79
RF	0.86	0.85	0.85	0.85
LR	0.88	0.84	0.87	0.87

Table A.2: Accuracy with different model and feature combinations using BoW encoding

Recall

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	0.57	0.47	0.76	0.69
CNN	0.00	0.00	0.00	0.00
LSTM	0.00	0.00	0.00	0.00
CNN-GRU	0.61	0.18	0.20	0.17
CNN-LSTM	0.07	0.01	0.00	0.01
RF	0.35	0.33	0.27	0.28
LR	0.45	0.25	0.43	0.43

Table A.3: Recall with different model and feature combinations using BoW encoding

Precision

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	0.7	0.7	0.62	0.72
CNN	0.00	0.00	0.00	0.00
LSTM	0.00	0.00	0.00	0.00
CNN-GRU	0.68	0.62	0.61	0.45
CNN-LSTM	0.5	0.08	0.00	0.08
RF	0.9	0.93	0.95	0.96
LR	0.9	0.85	0.87	0.87

Table A.4: Precision with different model and feature combinations using BoW encoding

F1 Score

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	0.61	0.55	0.67	0.7
CNN	0.00	0.00	0.00	0.00
LSTM	0.00	0.00	0.00	0.00
CNN-GRU	0.63	0.26	0.27	0.24
CNN-LSTM	0.12	0.01	0.00	0.02
RF	0.51	0.49	0.42	0.43
LR	0.6	0.39	0.58	0.58

Table A.5: F1 score with different model and feature combinations using BoW encoding

Time

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
FNN	49	42	44	43
CNN	917	1362	945	955
LSTM	97648	95868	105089	98019
CNN-GRU	7577	8046	8703	8055
CNN-LSTM	21656	22826	23481	22394
RF	12	51	23	23
LR	2	2	2	2

Table A.6: Execution time (seconds) with different model and feature combinations using BoW encoding

Results with Label encoding

Loss

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	0.53	0.55	0.38	0.40
LSTM	0.51	0.44	0.68	0.37
CNN-GRU	0.59	0.57	0.61	0.65
CNN-LSTM	0.56	0.50	0.59	0.58

Table A.7: Loss with different model and feature combinations using Label encoding

Accuracy

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	0.87	0.87	0.87	0.88
LSTM	0.87	0.88	0.88	0.88
CNN-GRU	0.86	0.82	0.86	0.88
CNN-LSTM	0.83	0.82	0.87	0.88

Table A.8: Accuracy with different model and feature combinations using Label encoding

Recall

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	0.62	0.60	0.68	0.75
LSTM	0.56	0.64	0.70	0.74
CNN-GRU	0.57	0.25	0.59	0.62
CNN-LSTM	0.33	0.41	0.68	0.66

Table A.9: Recall with different model and feature combinations using Label encoding

Precision

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	0.72	0.71	0.69	0.65
LSTM	0.68	0.70	0.69	0.66
CNN-GRU	0.65	0.67	0.67	0.74
CNN-LSTM	0.67	0.59	0.69	0.71

Table A.10: Precision with different model and feature combinations using Label encoding

F1 Score

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	0.64	0.63	0.68	0.69
LSTM	0.60	0.65	0.69	0.69
CNN-GRU	0.60	0.35	0.62	0.66
CNN-LSTM	0.42	0.47	0.68	0.67

Table A.11: F1 score with different model and feature combinations using Label encoding

Time

Model	Feature Comb. 1	Feature Comb. 2	Feature Comb. 3	Feature Comb. 4
CNN	51	54	141	151
LSTM	2,464	2,543	11731	14363
CNN-GRU	226	122	988	1169
CNN-LSTM	546	520	2643	3182

Table A.12: Execution time (seconds) with different model and feature combinations using Label encoding

Appendix B

Appendix: Result tables of phase 2

Loss

Model	Word2Vec	FastText
CNN	0.59	0.48
LSTM	0.34	0.31
CNN-GRU	0.39	0.39
CNN-LSTM	0.57	0.59

Table B.1: Loss with different model and word embedding

Accuracy

Model	Word2Vec	FastText
CNN	0.83	0.86
LSTM	0.87	0.91
CNN-GRU	0.88	0.89
CNN-LSTM	0.87	0.86

Table B.2: Accuracy with different model and word embedding

Recall

Model	Word2Vec	FastText
CNN	0.66	0.66
LSTM	0.54	0.64
CNN-GRU	0.47	0.55
CNN-LSTM	0.5	0.63

Table B.3: Recall with different model and word embedding

Precision

Model	Word2Vec	FastText
CNN	0.52	0.6
LSTM	0.67	0.83
CNN-GRU	0.75	0.77
CNN-LSTM	0.65	0.59

Table B.4: Precision with different model and word embedding

F1 Score

Model	Word2Vec	FastText
CNN	0.57	0.62
LSTM	0.59	0.71
CNN-GRU	0.57	0.62
CNN-LSTM	0.55	0.6

Table B.5: F1 score with different model and word embedding

Time

Model	Word2Vec	FastText
CNN	49	48
LSTM	128	126
CNN-GRU	31	34
CNN-LSTM	51	58

Table B.6: Execution time (seconds) with different model and word embedding

Appendix C

Appendix: Result tables of phase 3

Loss

Model	Count Encoding	TF-IDF
FNN	0.29	0.45
CNN-GRU	0.54	0.52

Table C.1: Loss with different model and BoW encoding algorithm

Accuracy

Model	Count Encoding	TF-IDF
FNN	0.90	0.87
CNN-GRU	0.88	0.79
LR	0.88	0.84

Table C.2: Accuracy with different model and BoW encoding algorithm

Recall

Model	Count Encoding	TF-IDF
FNN	0.69	0.65
CNN-GRU	0.61	0.00
LR	0.48	0.28

Table C.3: Recall with different model and BoW encoding algorithm

Precision

Model	Count Encoding	TF-IDF
FNN	0.72	0.68
CNN-GRU	0.68	0.00
LR	0.90	0.86

Table C.4: Precision with different model and BoW encoding algorithm

F1 Score

Model	Count Encoding	TF-IDF
FNN	0.70	0.65
CNN-GRU	0.63	0.00
LR	0.60	0.43

Table C.5: F1 score with different model and BoW encoding algorithm

Time

Model	Count Encoding	TF-IDF
FNN	43	41
CNN-GRU	7577	7834
LR	2	2

Table C.6: Execution time (seconds) with different model and BoW encoding algorithm

Appendix D

Appendix: Result tables of phase 4

Loss

Model	Not weighted	Weighted
FNN	0.29	0.32
CNN-GRU	0.54	0.51
LSTM	0.31	0.34

Table D.1: Loss with different model and label weights

Accuracy

Model	Not weighted	Weighted
FNN	0.90	0.91
CNN-GRU	0.88	0.90
LR	0.88	0.91
LSTM	0.91	0.90

Table D.2: Accuracy with different model and label weights

Recall

Model	Not weighted	Weighted
FNN	0.69	0.69
CNN-GRU	0.61	0.68
LR	0.45	0.61
LSTM	0.64	0.75

Table D.3: Recall with different model and label weights

Precision

Model	Not weighted	Weighted
FNN	0.72	0.79
CNN-GRU	0.68	0.74
LR	0.90	0.84
LSTM	0.83	0.70

Table D.4: Precision with different model and label weights

F1 Score

Model	Not weighted	Weighted
FNN	0.70	0.73
CNN-GRU	0.63	0.70
LR	0.60	0.71
LSTM	0.71	0.72

Table D.5: F1 score with different model and label weights

Time

Model	Not weighted	Weighted
FNN	43	51
CNN-GRU	7577	8134
LR	2	2
LSTM	126	231

Table D.6: Execution time (seconds) with different model and label weights

Appendix E

Appendix: Result tables of phase 5

Loss

Model	500	1000	2000	4000	6000
FNN	0.49	0.48	0.40	0.32	0.32
CNN-GRU	0.69	0.77	0.52	0.55	0.56
LSTM	0.50	0.45	0.41	0.32	0.29

Table E.1: Loss with different model and data set sizes

Accuracy

Model	500	1000	2000	4000	6000
FNN	0.82	0.84	0.86	0.90	0.91
CNN-GRU	0.81	0.83	0.86	0.89	0.91
LR	0.84	0.86	0.88	0.90	0.91
LSTM	0.81	0.84	0.87	0.88	0.89

Table E.2: Accuracy with different model and data set sizes

Recall

Model	500	1000	2000	4000	6000
FNN	0.54	0.61	0.66	0.73	0.76
CNN-GRU	0.48	0.53	0.60	0.62	0.66
LR	0.27	0.32	0.47	0.55	0.61
LSTM	0.61	0.66	0.63	0.72	0.74

Table E.3: Recall with different model and data set sizes

Precision

Model	500	1000	2000	4000	6000
FNN	0.50	0.55	0.61	0.70	0.74
CNN-GRU	0.45	0.52	0.59	0.73	0.81
LR	0.65	0.72	0.80	0.83	0.84
LSTM	0.48	0.55	0.63	0.64	0.69

Table E.4: Precision with different model and data set sizes

F1 Score

Model	500	1000	2000	4000	6000
FNN	0.50	0.57	0.62	0.71	0.75
CNN-GRU	0.46	0.52	0.58	0.67	0.72
LR	0.37	0.44	0.58	0.66	0.71
LSTM	0.53	0.60	0.62	0.67	0.70

Table E.5: F1 score with different model and data set sizes

Time

Model	500	1000	2000	4000	6000
FNN	3	3	4	7	9
CNN-GRU	247	291	327	392	441
LR	1	1	1	2	2
LSTM	63	127	154	193	227

Table E.6: Execution time (seconds) with different model and data set sizes

Appendix F

Appendix: Result tables of phase 6

F.1 Part 1

Loss

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	0.58	0.46	0.53	0.30	0.75	0.36	0.62
CNN-GRU	0.52	0.44	0.37	0.40	0.75	0.34	-

Table F.1: Loss with different model and independent features

Accuracy

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	0.80	0.92	0.92	0.91	0.18	0.93	0.83
CNN-GRU	0.72	0.91	0.91	0.89	0.18	0.90	-
LR	0.73	0.90	0.92	0.91	0.82	0.92	0.83

Table F.2: Accuracy with different model and independent features

Recall

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	0.71	0.74	0.73	0.66	1.00	0.71	0.06
CNN-GRU	0.74	0.78	0.77	0.70	1.00	0.72	-
LR	0.61	0.74	0.66	0.61	0.00	0.66	0.05

Table F.3: Recall with different model and independent features

Precision

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	0.47	0.79	0.80	0.82	0.18	0.86	0.42
CNN-GRU	0.36	0.74	0.73	0.71	0.18	0.71	-
LR	0.35	0.70	0.86	0.84	0.00	0.86	0.60

Table F.4: Precision with different model and independent features

F1 Score

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	0.55	0.76	0.75	0.73	0.30	0.77	0.09
CNN-GRU	0.48	0.75	0.75	0.70	0.30	0.71	-
LR	0.45	0.72	0.74	0.71	0.00	0.74	0.09

Table F.5: F1 score with different model and independent features

Time

Model	C1	C3	C5	W1	W2	W3	UP, !!, O? & O+
FNN	1	7	53	1	3	10	1
CNN-GRU	24	182	1284	164	474	972	-
LR	1	2	8	1	2	7	1

Table F.6: Execution time (seconds) with different model and independent features

F.2 Part 2

Loss

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	0.35	0.35	0.32	0.46	0.51	0.40

Table F.7: Loss with different model and combinations of features with C5

Accuracy

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	0.92	0.93	0.92	0.93	0.93	0.92
LR	0.91	0.92	0.93	0.92	0.92	0.92

Table F.8: Accuracy with different model and combinations of features with C5

Recall

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	0.79	0.79	0.72	0.69	0.69	0.68
LR	0.73	0.75	0.69	0.64	0.65	0.66

Table F.9: Recall with different model and combinations of features with C5

Precision

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	0.77	0.82	0.81	0.86	0.85	0.83
LR	0.77	0.81	0.87	0.87	0.88	0.87

Table F.10: Precision with different model and combinations of features with C5

F1 Score

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	0.77	0.79	0.75	0.76	0.75	0.73
LR	0.75	0.78	0.77	0.74	0.75	0.75

Table F.11: F1 score with different model and combinations of features with C5

Time

Model	C1	C3	W1	W2	W3	UP, !!, O? & O+
FNN	39	42	9	62	132	7
LR	14	15	4	22	71	3

Table F.12: Execution time (seconds) with different model and combinations of features with C5

F.3 Part 3

Loss

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	0.33	0.65	0.36	0.53	0.30

Table F.13: Loss with different model and combinations of features with C3 and C5

Accuracy

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	0.93	0.93	0.92	0.92	0.94
LR	0.91	0.93	0.92	0.92	0.93

Table F.14: Accuracy with different model and combinations of features with C3 and C5

Recall

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	0.86	0.66	0.68	0.70	0.79
LR	0.73	0.69	0.64	0.65	0.76

Table F.15: Recall with different model and combinations of features with C3 and C5

Precision

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	0.76	0.89	0.87	0.86	0.82
LR	0.77	0.87	0.87	0.88	0.81

Table F.16: Precision with different model and combinations of features with C3 and C5

F1 Score

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	0.79	0.75	0.75	0.76	0.80
LR	0.75	0.77	0.74	0.75	0.78

Table F.17: F1 score with different model and combinations of features with C3 and C5

Time

Model	C1	W1	W2	W3	UP, !!, O? & O+
FNN	48	47	54	63	42
LR	18	16	51	59	17

Table F.18: Execution time (seconds) with different model and combinations of features with C3 and C5

Appendix G

Appendix: Result table of phase 7

Model	Accuracy	Recall	Precision	F1
FNN	0.92	0.63	0.91	0.75
CNN-GRU	0.91	0.79	0.73	0.76
LSTM	0.89	0.75	0.68	0.71
LR	0.94	0.80	0.83	0.82
Ensemble DT	0.93	0.77	0.83	0.80
Ensemble NN	0.93	0.19	0.23	0.21

Table G.1: Performance using test set with different models