INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Towards Cyberbullying Detection On Social Media

Tiago Filipe Pardal de Almeida

Master's Degree in Computer Engineering

Supervisor:
PhD, Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
PhD, Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

November, 2021

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Towards Cyberbullying Detection On Social Media

Tiago Filipe Pardal de Almeida

Master's Degree in Computer Engineering

Supervisor:
PhD, Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor,
Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
PhD, Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

November, 2021

I dedicate this work to my family and my girlfriend
for all the love they give me every day!

Acknowledgment

Firstly, I would like to thank my supervisors, Professor Ricardo Ribeiro and Professor Fernando Batista for their guidance, support, and suggestions in all the stages of this work, despite the setbacks that occurred throughout this period.

A big thanks to my parents, Ana and José, and my brother, André, for all their unconditional support and compassion during this special journey. To my girlfriend, Daniela, a special thanks for motivating me to achieve my goals and for always believing in me, especially in the moments that I did not believe in myself. I love you all.

Thanks to all my friends for encouraging me to never give up, as well as for cheering me up in the hardest moments.

Thank you all so much, none of this would be possible without them.

Resumo

O contínuo aparecimento do cyberbullying nas redes sociais constitui um problema mundial que tem aumentado consideravelmente nos últimos anos, e exige medidas urgentes para a deteção automática de tal fenómeno. O objetivo deste trabalho é criar um modelo suficientemente capaz de detetar automaticamente textos ofensivos. Para tal, foram utilizados três conjuntos de dados públicos, bem como duas abordagens principais para resolver este problema: uma baseada em métodos clássicos de Aprendizagem Automática e a outra baseada em Aprendizagem Profunda.

Na abordagem clássica de Aprendizagem Automática foi proposta uma fase específica de pré-processamento e Engenharia de Características com várias etapas. Para além disso, foram exploradas duas abordagens de representação de documentos para gerar as entradas utilizadas pelos classificadores SVM, Logistic Regression e Random Forest. Uma vez que estes conjuntos de dados são desequilibrados, SMOTEENN e Threshold-Moving foram utilizados para lidar com o problema de classificação desbalanceada.

Na abordagem de Aprendizagem Profunda foram exploradas diferentes arquiteturas, combinando vetores de palavras pré-treinados com CNN, CNN-Attention, BiLSTM e BiLSTM-Attention. A configuração experimental envolveu o tratamento de palavras desconhecidas, *Cyclical Learning Rate* para proporcionar uma melhor convergência, Macro Soft-F1 Loss para otimizar o desempenho e Macro Soft-F2 Loss para lidar com o problema de classificação desbalanceada. Foi também proposto um modelo RoBERTa-base, pré-treinado em 58 milhões de tweets e afinado para identificação de linguagem ofensiva.

Os resultados experimentais mostram que, embora seja uma tarefa difícil, ambas as abordagens propostas são adequadas para detetar textos ofensivos. No entanto, a abordagem de Aprendizagem Profunda alcança os melhores resultados.

Palavras-chave: redes sociais, linguagem ofensiva, representação de palavras, Engenharia de Características, Aprendizagem Profunda

Abstract

The continuous appearance of cyberbullying on social media constitutes a worldwide problem that has seen a considerable increase in recent years, and demands urgent measures to automatically detecting such phenomenon. The goal of this work is to create a model sufficiently capable of automatically detecting offensive texts. For this purpose, three public datasets were used, as well as two main approaches to solve this problem: one based on classical Machine Learning methods and the other based on Deep Learning.

In the classical Machine Learning approach was proposed a specific pre-processing and Feature Engineering stage with several steps. In addition, two document representation approaches were also explored to generate the inputs used by SVM, Logistic Regression, and Random Forest classifiers. Since these datasets are imbalanced, SMOTEENN and Threshold-Moving were used to deal with the imbalanced classification problem.

In the Deep Learning approach different architectures were explored, combining pre-trained word vectors with CNN, CNN-Attention, BiLSTM and BiLSTM-Attention. The experimental setup involved treatment of unknown words, Cyclical Learning Rate to provide better convergence, Macro Soft-F1 Loss function to optimize performance and Macro Soft-F2 Loss function to deal with the imbalanced classification problem. RoBERTa-base model was also proposed, pre-trained on 58 million tweets and fine-tuned for offensive language identification.

Experimental results show that, although it is a difficult task, both proposed approaches are suitable for detecting offensive texts. Nevertheless, the Deep Learning approach achieves the best results.

Keywords: social media, offensive language, word representation, Feature Engineering, Deep Learning

Abbreviations

ADASYN: Adaptive Synthetic
AI: Artificial Intelligence
ALBERT: A Lite Bidirectional Encoder Representations from Transformers
BERT: Bidirectional Encoder Representations from Transformers
BiGRU: Bidirectional Gated Recurrent Unit
BiLSTM: Bidirectional Long Short-Term Memory
BoW: Bag-of-Words
BPE: Byte-Pair Encoding
CBOW: Continuous Bag-Of-Words
CLR: Cyclical Learning Rate
CNN: Convolutional Neural Network
COVID-19: Coronavirus Disease 2019
CRISP-DM: Cross-Industry Standard Process for Data Mining
DNN: Deep Neural Network
ELMo: Embeddings from Language Models
FN: False Negatives
FP: False Positives
GloVe: Global Vectors for Word Representation
GPT-2: Generative Pre-trained Transformer 2
GPU: Graphics Processing Unit
GRU: Gated Recurrent Unit
IDF: Inverse Document Frequency
KNN: K-Nearest Neighbors
L-BFGS: Limited-memory Broyden–Fletcher–Goldfarb–Shanno
LDA: Latent Dirichlet Allocation
LGBT: Lesbian, Gay, Bisexual and Transgender
LSI: Latent Semantic Indexing
LSTM: Long Short-Term Memory
mBERT: Multilingual BERT
MTL: Multi-Task Learning
Newton-CG: Newton-Conjugate Gradient
NLP: Natural Language Processing
NLTK: Natural Language ToolKit
NN: Neural Network

OLID: Offensive Language Identification Dataset
OOV: Out-of-Vocabulary
POS: Part-of-Speech
RBF: Radial Basis Function
ReLU: Rectified Linear Unit
RNN: Recurrent Neural Network
RoBERTa: Robustly Optimized BERT Pretraining Approach
ROC: Receiver Operating Characteristic
SAG: Stochastic Average Gradient
SMOTE: Synthetic Minority Oversampling TEchnique
SMOTEENN: Synthetic Minority Oversampling Technique Edited Nearest Neighbours
SOLID: Semi-Supervised Offensive Language Identification Dataset
SSWE: Sentiment-Specific Word Embeddings
SVM: Support Vector Machines
TF: Term Frequency
TF-IDF: Term Frequency–Inverse Document Frequency
TN: True Negatives
TP: True Positives
VADER: Valence Aware Dictionary and sEntiment Reasoner
XLM-RoBERTa: Cross-Lingual Language Robustly Optimized BERT Pretraining Approach

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter is organized as follows. The motivation and context are introduced in Section 1.1, giving an overview of this work. The description of the research questions in Section 1.2, the goals in Section 1.3, and the research methodology in Section 1.4 are also discussed in this chapter. At the end of the chapter, the structure of the document is presented in Section 1.5.

1.1. Motivation and Context

With the growth of the Internet on a large scale, more sophisticated means of communication emerged. Social media such as Twitter, Facebook, and Instagram, among others, have a great impact on social relationships, especially among teenagers. Despite these platforms facilitate communication, many ethical issues have arisen like cyberbullying, which is analogously related to offensive language and hate speech, and is a relevant problem rooted in society.

Various definitions of cyberbullying are based on the traditional bullying criteria: the intent to inflict harm on the victim, the repetition of the behavior over time, and an imbalance of power between the victim and the bully. Traditional bullying is usually defined as being an aggressive, intentional act or behaviour that is carried out by a group or an individual repeatedly and over time against a victim who can not easily defend him or herself [1, 2]. StopBullying.gov addresses cyberbullying as "*bullying that takes place over digital devices like cell phones, computers, and tablets. Cyberbullying can occur through SMS, Text, and apps, or online in social media, forums, or gaming where people can view, participate in, or share content. Cyberbullying includes sending, posting, or sharing negative, harmful, false, or mean content about someone else. It can include sharing personal or private information about someone else causing embarrassment or humiliation*" [3].

A report published in 2013 by the anti-bullying charity "Ditch the Label", based on a sample of 10008 young people, revealed that 2/3 of the surveyed people aged 13–22 years old have been victims of cyberbullying [4]. According to Nalini and Jaba Sheela [5], Facebook, YouTube and Twitter are the most common networks for cyberbullying, as 54%, 21% and 28% of their users have suffered cyberbullying respectively. A recent study by Iscte – Instituto Universitário de Lisboa with the participation of 485 students of basic, secondary and higher education from all districts of Portugal concludes that more than 60% of the students were victims of cyberbullying in more than one situation during the lockdown caused by the Coronavirus Disease 2019 (COVID-19) pandemic, from March to

May of 2020. The main targets of the attacks turned out to be Lesbian, Gay, Bisexual and Transgender (LGBT) students and those with low family incomes. The aggressors indicated feeling indifference, anger and joy, but only 16% admitted feeling guilt and 41% of them assumed to do it "for fun", "for revenge on other episodes that happened" or "because they wanted to assert themselves" [6]. Victims of cyberbullying tend to suffer several negative effects, such as low self-esteem, mental depression, suicide consideration and even commit suicide [7]. Concerning suicide, these victims are 2 to 9 times more likely to commit suicide than non victims [8]. Social media texts associated with cyberbullying can be classified according to the language used as *explicit* or *implicit* expressions. Explicit expressions take place when using profane words that have a negative sentiment, while implicit expressions are associated with ironic or sarcastic expressions that do not have foul words [9].

Research about the detection of cyberbullying, hate speech, and offensive language in online texts has been carried out over the years, but these tasks are still considered a recent practice. Although cyberbullying has received increasingly attention from the social sciences research perspective, it has received less attention from the Natural Language Processing (NLP) and Machine Learning research perspectives [10]. Due to all these circumstances, this type of problems requires an effective response and therefore, the creation of automatic detection systems has become an extremely important task. These systems help to identify offensive texts, which can be a useful tool to avoid these problems and minimize their occurrence. By nature, this problem is an imbalanced classification problem, existing many more texts labeled as having no offensive content than as having offensive content. NLP techniques have been successfully applied to many text classification problems, and more recently to detect portions of texts related to the cyberbullying phenomena. Several methods have already been applied, ranging from Feature Engineering and rule-based systems to Machine Learning models. The most recent literature on this subject reveals that Deep Learning methods have been increasingly adopted to this task, due to their tendency to achieve improved performance.

1.2. Research Questions

In this section, I introduce the research questions, which will guide the work and are answered at the end of this document. These research questions are the basis of this work and through the answers it is possible to produce clearer conclusions. The research questions are as follows:

- Research question 1: How does classical Machine Learning methods compare with Deep Learning approaches in the detection of offensive language on social media?
- Research question 2: Are the approaches applied in this work suitable for various datasets?

The intent of the first research question is to explore classical Machine Learning methods and Deep Learning approaches in the detection of offensive language on social media, in order to compare and understand these approaches.

To answer the second research question it is necessary to test these approaches for various datasets, and find out whether they are suitable for all datasets, only for some, or for none.

1.3. Goals

This type of problem dramatically affects many children and adolescents, and has serious negative consequences. This is the main reason and the starting point for the proposed goals, in which each goal has an influence on the next one. The proposed goals are as follows:

- Goal 1: Identify the most commonly used methods and models for this type of detection;
- Goal 2: Develop effective methods to help models achieve good performance;
- Goal 3: Develop automatic detection models to identify offensive texts;
- Goal 4: Evaluate the ability of the models to correctly classify.

The main focus in the first goal is to identify the methods and models already used in this field and understand the impact of these methods on the development of an efficient model. This goal is important as it makes it possible to find out which methods and models are the most appropriate to develop an effective solution.

The second goal focuses on the development of the most relevant methods identified in the first goal, as well as other important methods. The performance of the models can be leveraged by using these methods, thus allowing better results to be achieved.

The third goal involves the development of automatic detection models. These models are fed by inputs, which in this case are text representations, identifying each text as being offensive or not.

The fourth goal aims to evaluate the performance of the models through evaluation metrics. Models such as these can identify these problems on social media, helping to reduce their occurrence, which have been increasing dramatically over the past few years.

1.4. Research Methodology

The research methodology followed in this work is the Cross-Industry Standard Process for Data Mining (CRISP-DM), which is a comprehensive data mining methodology and a standard methodology applied to data knowledge extraction. This methodology is highly proven and provides a structured approach to the planning and development phase of the project [11, 12]. Since the ultimate goal is to develop a predictive model that allows the automatic identification of offensive texts, this methodology is the most appropriate to be used in this work. CRISP-DM breaks down the life cycle of a project into six steps, it is iterative and may not be sequential. These stages can be performed countless

times, depending on the results of the evaluation. All CRISP-DM steps are illustrated in Figure 1.1.

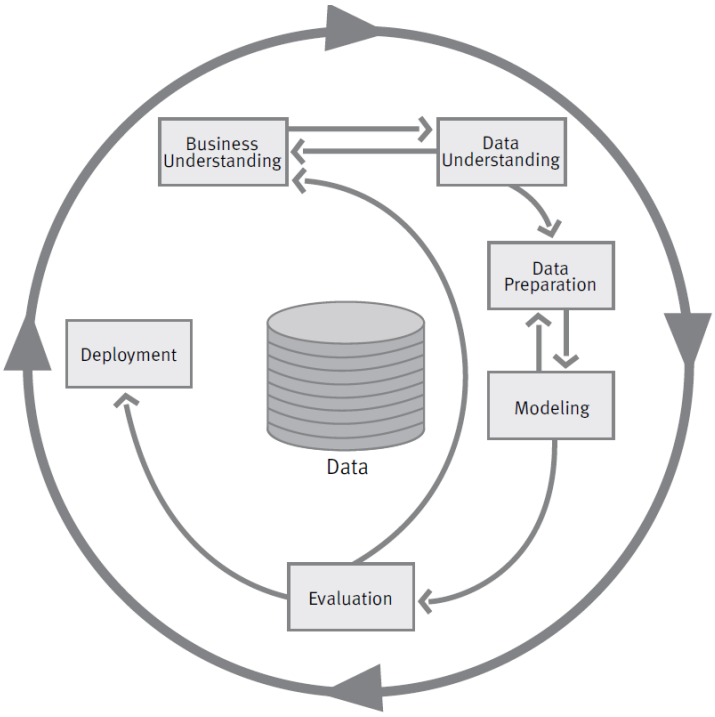


FIGURE 1.1. Steps of CRISP-DM [13].

The first step is called Business Understanding being attached to Chapter 1 and Chapter 3 because it can be interpreted as the step of analysis, understanding, definition, contextualization, determination of goals and plan production of the current problem. The identification and understanding of the most used methods in this case is also addressed.

The next step is the Data Understanding, which can be seen in Chapter 4, intended for the acquisition, description, analysis, and verification of data quality. It may be necessary to return to Business Understanding if doubts arise concerning the data, since it is important to understand well the theoretical and complete scope of the problem to understand the data equally well. In this work several datasets were used with the intention of understanding the generalization of the methods applied and whether they are really effective in most of cases.

The third step is the Data Preparation, a fundamental step to achieve better results, where Text Mining techniques are applied. Data cleaning and pre-processing are the most relevant tasks at this step, culminating in better data structuring. This step is further addressed in Section 4.2.

In this work, the Modelling step is presented in Chapter 5 and consists in the application of NLP with several learning techniques, where the algorithms receive inputs and return the respective predictions, allowing the creation of the automatic detection models. During this step it is very frequent to return to Data Preparation, since features with great relevance or little relevance to the models may have been identified, increasing the tasks in Data Preparation in order to achieve better results.

The results of the models are evaluated and analyzed with the aim of finding out if it is necessary to revisit the previous steps to improve the results and check whether the results of the models are in accordance with the proposed goals, being this step called Evaluation and being described in Chapter 6. This step can be aggregated to the Deployment step that is destined to choose the most suitable model according to the characteristics of the problem, results, and proposed goals.

1.5. Document Structure

The structure of this document is described as follows:

- Chapter 2: A set of fundamental concepts important to better understand the topics covered in this work are presented and explained in this chapter;
- Chapter 3: A literature review of relevant works related to the automatic detection of cyberbullying, hate speech, and offensive language in social media texts is performed;
- Chapter 4: The datasets used in this work are presented and described. The preparation of the texts is also addressed in this chapter, allowing to achieve improvements in the performance of the models, through Text Mining techniques, refined cleaning, and pre-processing of data;
- Chapter 5: The process of developing the various models is detailed in this chapter. The approaches implemented to deal with overfitting and the techniques to achieve better and more stable performance are also explained;
- Chapter 6: The evaluation of the models performance is reported and discussed using the results;
- Chapter 7: The conclusions and limitations encountered in the course of the work, as well as future work are discussed in this chapter.

CHAPTER 2

Fundamental Concepts

In this chapter, a general description of several important concepts is made, in order to allow a better understanding of the topics covered, which can be considered as a set of bases for the perception of the implementations carried out throughout this work. The concepts described in this chapter are as follows: a brief explanation of Feature Engineering in Section 2.1, basic concepts about Machine Learning in Section 2.2, Cross-Validation in Section 2.3, Deep Learning in Section 2.4, and word representation in Section 2.5.

2.1. Feature Engineering

Feature Engineering exploits domain knowledge to extract useful features from the data. Features are attributes that provide clues for the underlying problem, helping to solve it [14]. The creation and definition of these features depends on the purpose for which they are being used, which in this case is the classification of offensive texts. So, to create features, it is necessary to understand and have knowledge about the data content. These features, when well selected, combined, and applied, can be used to improve the performance of models. Basically, the features are the inputs consumed by the predictive models, being these features a representation of the data that helps the models learn a solution for a given problem.

2.2. Machine Learning

Machine Learning is a subfield of Artificial Intelligence (AI) concerning to systems that learn automatically based on knowledge without being directly programmed, being this knowledge coming from data. These systems learn from observations or data, looking for patterns, allowing to make the best decisions in the future based on the data previously provided. The following definition characterize Machine Learning in a few words "*The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience*" [15]. The same author provides the formalism "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*" [15].

Machine Learning approaches can be broadly categorized as Unsupervised Learning or Supervised Learning. Unsupervised Learning is defined as a type of learning based on patterns from unlabelled data data. Clustering is other type of Unsupervised Learning, which consists of splitting the data into clusters of related instances. Supervised Learning algorithms learn from the labeled data, in which each instance of this data presents a label and target, contrarily to Unsupervised Learning. These algorithms are described as

constructors of a mathematical model of a dataset that contains both the inputs and the desired outputs. There are other variants of the learning paradigm, like Semi-Supervised Learning, whereby some instances include a supervision target but others do not, or Reinforcement Learning, where algorithms or agents interact and are exposed to an environment, learning from experience. I focused on Supervised Learning because the data used in this work are labelled data.

Supervised Learning problems can be grouped into Regression and Classification problems and both problems aim to build a succinct model that can predict the value of the dependent attribute from the attribute variables, with the task T being associated with these problems. In the Classification task, the algorithms produce the following function (Equation 2.1), with k classes or categories:

$$f : \mathbf{R}^n \to \{1, 2, \ldots, k\} \tag{2.1}$$

In this case, the algorithms have the mission to specify to which of the k classes the inputs belong. It is better understood with the approximation function (Equation 2.2), where the goal is to predict the output y given new inputs x:

$$y = f(x) \tag{2.2}$$

In contrast, regression algorithms produce the following function (Equation 2.3):

$$f : \mathbf{R}^n \to \mathbf{R} \tag{2.3}$$

The main difference between these two tasks is that the dependent attribute is categorical for Classification and numerical for Regression. In this work, I address a binary classification problem, since there are only two distinct classes: non-offensive class (negative class) and offensive class (positive class).

Performance measures or evaluation metrics are measures to help evaluate the quality of models. There are evaluation metrics for the several types of problems described above, but only the classification metrics are addressed in this work. These evaluation metrics are described in Section A.

Typically, the data is separated into three distinct sets: training set, validation set, and testing set. Initially, Machine Learning models have access only to the training set in order to learn from this data, and later they also have access to the testing set, which allows them to check their performance for this data. The main challenge in Machine Learning is to create conditions for algorithms to perform well for new inputs that are outside the data universe where they were trained. This ability is called generalization and can be verified with the validation set. There are two big challenges aggregated to the generalization: underfitting and overfitting. Underfitting occurs when a model neither models the training set well nor generalises well to new data. Overfitting occurs when

a model models the training set too well, while performing poorly on the validation set. The basic concepts about classical Machine Learning algorithms, namely Support Vector Machines (SVM), Logistic Regression and Random Forest are presented in Section B.

2.3. Cross-Validation

Cross-Validation is a statistical method that is used to assess the generalization of a model, in other words, its performance for a new dataset. There are several types of Cross-Validation, such as Holdout, k-Fold, Stratified k-Fold, among others. The most frequently used Cross-Validation method is the k-Fold Cross-Validation. With k-Fold Cross-Validation, the training dataset is split into k equal partitions, with one of these partitions designated as the validation set and the remaining partitions as the training set. This procedure is repeated for k folds and for each fold, the partition named as the validation set is always different from the partition of the previous folds and consequently the partitions selected as the training set are also different from the previous ones. At each fold, the model is trained with the training set, tested with the validation set and the validation score is stored. At the end of all folds, it is calculated the average of the stored validation scores, thus producing the final score average. Figure 2.1 represents an example of k-Fold Cross-Validation, with k=4:

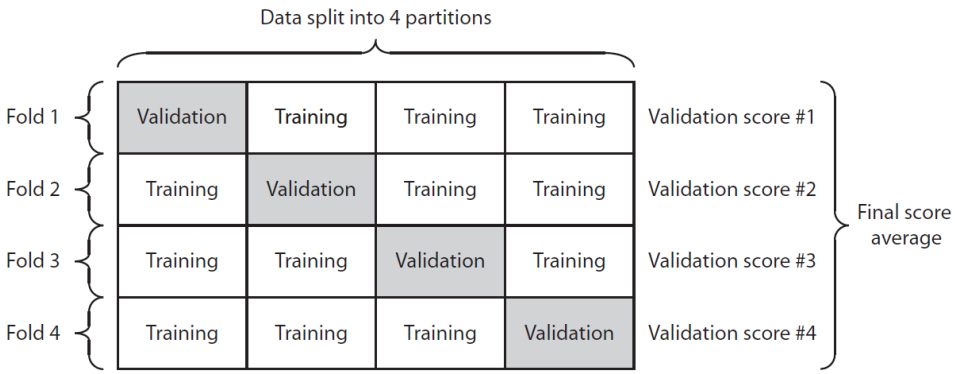


FIGURE 2.1. 4-Fold Cross-Validation [16].

2.4. Deep Learning

Although Deep Learning has been around since the 1940s, it was only in the last decade that it began to be used on a large scale, having undergone important scientific advances in several areas, namely Computer Vision, Automatic Speech Recognition, NLP, Audio Recognition, and Bioinformatics, producing state-of-the-art results and a huge scientific evolution. Deep Learning has been a core part of AI, and a subset of Machine Learning, which can be understood as a collection of algorithms and techniques inspired by the way human brain operates in data processing and creation of patterns for use in decision making, being called Deep Neural Network (DNN). The DNN is composed of connected units called artificial neurons that are aggregated into several layers. The DNN also maps inputs to targets, but unlike Machine Learning, it is through the deep sequence of

simple data transformations. The transformation that each layer performs on the inputs is parameterized by its weights, or also known as parameters, which are represented by numbers and the specifications of what each layer implements. The goal of this approach is to find the most appropriate weights w for all layers in a DNN, allowing the inputs x to be mapped correctly to their associated targets, producing the outputs or predictions y. This goal is represented by the function (Equation 2.4):

$$y = f(x, w) \tag{2.4}$$

An illustration of this goal is shown in Figure 2.2:

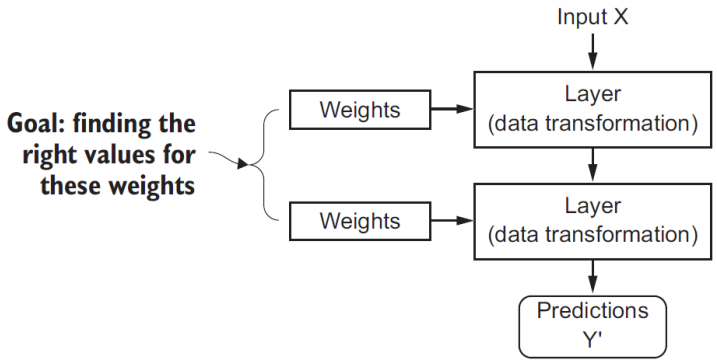


FIGURE 2.2. The goal of Deep Neural Networks [17].

This goal is very exhaustive and hard because the DNN can contain tens of millions of weights and changing one parameter will affect the behavior of all the others. To measure the distance between the predictions of the DNN and the true targets is used the objective function or cost function or better known as the loss function. Loss function has the mission to take the predictions and true targets, calculate a distance score between them and produce a loss score or error. If the loss scores are falling, it means that the values of predictions and true targets are getting closer and the DNN is learning ever better. The loss score is used as a feedback signal to slightly adjust the weights in all layers and this adjustment is realized by the optimizer. At the beginning of the training, random weights are assigned to the layers, which provide a high loss score, but by repeating this process a sufficient number of times, the loss score starts to decrease. This process can be observed in Figure 2.3.

The goal in this step is to learn the function f that minimizes the expected loss score provided by the loss function (Equation 2.5):

$$L(f(x), y) \tag{2.5}$$

Basically, the DNN tries to find the function f that approximates the Equation 2.6, where the w is the learned weights and w' the adjusted weights:

$$w' \approx \underset{w}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; w), y_i) \tag{2.6}$$

This area of learning has been achieving extraordinary advances in text classification, having overcome classical Machine Learning approaches in several tasks. In this area, Word Embeddings and the correct choice of pre-trained word embeddings models are fundamental to achieve good performances. The basic concepts about Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BiLSTM) are presented in Section C.

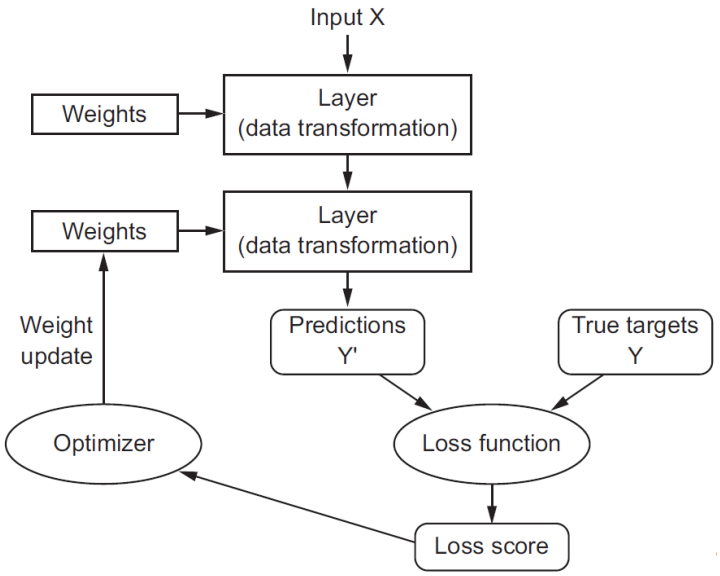


FIGURE 2.3. The training process in Deep Neural Networks [17].

2.5. Word Representation

Classical methods to represent words, such as One-Hot Encoding, Bag-of-Words (BoW) and Term Frequency–Inverse Document Frequency (TF-IDF) are widely used in this area. The most basic method is One-Hot Encoding, in which each word is represented as a unique one-hot vector containing 1 and 0. The representation of the document collection results in a sparse matrix, where columns are terms and rows are documents. BoW is a simple approach that allows to represent sentences in vectors with the frequency of the words that occur in these sentences. TF-IDF is a statistical method, allowing to find the importance of each word for a document, considering a collection of documents, and it is represented by the multiplication between Term Frequency (TF) and Inverse Document Frequency (IDF):

$$\text{TF-IDF} = tf_{t,d} \times idf_t \tag{2.7}$$

TF is a weight that represents the number of times a term t appears in a document d and is usually calculated by the logarithmic frequency:

$$tf_{t,d} = 1 + \log(f_{t,d}) \tag{2.8}$$

IDF is a weight that represents how commonly a term t appears in the documents, the more often a term appears in documents, the less weight and less importance it has. It is calculated as follows, being the N the number of documents and df_t the number of documents that contain the term t:

$$idf_t = \log(\frac{N}{df_t}) \tag{2.9}$$

Word2vec and Global Vectors for Word Representation (GloVe) are Word Embedding techniques that have the capacity to better capture the semantic and syntactic interaction between words, as well as the meaning of words in a document. Word Embeddings is one of the key breakthroughs for the impressive performance of Deep Learning methods, being defined as a dense representation of words in the text. These words are mapped to vectors of real numbers, being a learned representation for text where words that have the same meaning have a similar representation. These models were created one year apart: Word2vec [18, 19] was created in 2013 and since then has become the development standard for the pre-trained word embeddings, while GloVe [20] was created in 2014. GloVe is based on the Word2Vec prerequisites, so both are word-based and context independent models, and generate only one embedding for each word, integrating all the various meanings of the word into a single vector.

Word2vec takes the input from a large text corpus and generates the output to vector space, in which each word vector is put in this vector space, whereas these vectors reflect the degree of semantic correlation between the words represented by these vectors. This model gets the relationships of words with the aid of window size using two types of learning models: Skip-Gram and Continuous Bag-Of-Words (CBOW). Both models learn the underlying word representations for each word by using DNN. Skip-Gram and CBOW are illustrated in Figure 2.4.

In the Skip-Gram method, it takes into account the center word as input and the neighbouring words as output, predicting the neighbouring words based on this center word. The size of this window is a configurable parameter of the model and represents the number of neighboring words. It fits best with a small dataset and very well distinguishes unusual words. The main goal is to calculate the following probability distribution, where T is the size of the corpus, -l and l are the limits of window, and word w_i is each word of the corpus:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-l\leqslant j\leqslant l}\log P(w_{i+j} \mid w_i) \tag{2.10}$$

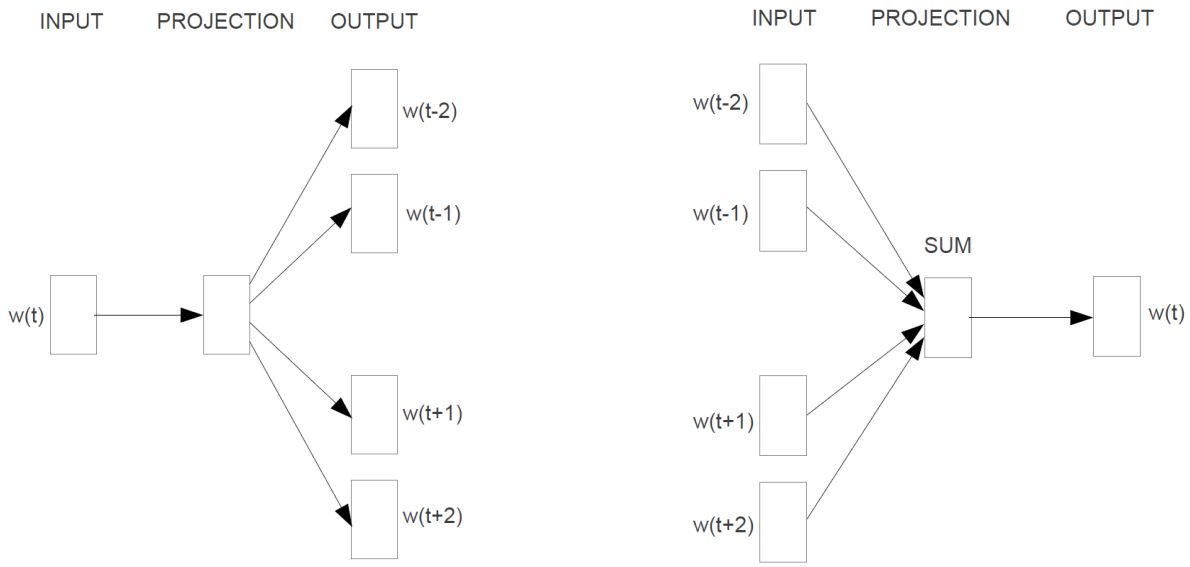


FIGURE 2.4. Skip-Gram (left) and CBOW (right)

The first step is get the hidden or projection as follows, where M is the weights matrix and x is the input vector:

$$h = M^T x \tag{2.11}$$

Having obtained the vector h from the projection, the next step is to compute the output by calculating d multinomial distribution, with v' as output vector of the word w with index i, for each entry u with index i:

$$u_{d,i} = {v'_{w_i}}^T h \tag{2.12}$$

Since the output activation is softmax, the output is calculated as follows, where the word w with index j is the projection:

$$P(w_{d,i} = w_{0,c} \mid w_j) = y_{c,i} = \frac{e^{(u_{d,i})}}{\sum_{i'=1}^{V} e^{(u_{i'})}} \tag{2.13}$$

In relation to the CBOW method, this method is simply an inverse of the Skip-Gram method, taking the neighbouring words as inputs and the center word as output. It works good with large datasets and can create better representation for frequent words. The goal of this method is to calculate the following probability distribution:

$$P(w_0 \mid w_{1,1}, \ldots, w_{i,c}) \tag{2.14}$$

The projection is computed as follows, where the average of input vectors x ranges from 1 to c:

$$h = \frac{1}{c} W(x_1 + x_2, \ldots, x_c) \tag{2.15}$$

The cost function has the following form:

$$-\log(P(w_0 \mid w_{1,1}, \ldots, w_{i,c})) \tag{2.16}$$

The output without activation is calculated in a similar way to the Skip-Gram method:

$$u_i = {v'_{w_i}}^T h \tag{2.17}$$

The output is calculated with softmax as activation as follows:

$$P(w_i \mid w_j) = y_i = \frac{e^{(u_i)}}{\sum_{i'=1}^{V} e^{(u_{i'})}} \tag{2.18}$$

On the other hand, GloVe captures the meaning of a word with the structure of the entire text corpus, building a word co-occurrence matrix using statistics, i.e., minimizing the least-squares error and at the end produces a word vector space. To do this, it is necessary to compute the following loss function, where the word with index i occurs in the context of the word j, with biases b and weighting function f, using the co-occurrence matrix M:

$$\sum_{i,j=1}^{V} f(M_{ij})(w_i^T w_j + bi + bj - \log(M_{ij}))^2 \tag{2.19}$$

GloVe has the advantage of using word vectors to catch sub-linear relationships in vector space.

The most common use of Word Embeddings is the use of pre-trained word embeddings models which are embeddings trained on vast datasets, stored and then used to solve other tasks, being a form of Transfer Learning. Transfer Learning is a method where a model developed for a task is reused for a new problem and it is currently very popular in Deep Learning. The embeddings from the pre-trained word embeddings models are used as initialization weights instead of initialize the weights randomly.

CHAPTER 3

Literature Review

This chapter focuses on the literature review related to the detection of cyberbullying, hate speech, and offensive language on social media texts. Section 3.1 focuses on the classical approaches, Section 3.2 focuses on the Deep Learning approaches, and Section 3.3 presents a brief summary of the chapter.

3.1. Classical Approaches

Classical approaches have been successfully exploited by researchers in text-based classification problems, with Feature Engineering being the most common approach to detect cyberbullying, using domain knowledge to create additional features from the data being exploited. Following this approach, Dadvar et al. [21] proposed a system that uses a set of features and SVM as classifier to detect cyberbullying on YouTube comments. Stop words removal and Stemming were applied as pre-processing and they reported that content-based features and user-based features improve the detection of cyberbullying. Dadvar et al. [22] developed a gender-specific text classifier using SVM and the MySpace texts provided by Fundaction Barcelona Media, demonstrating that gender-specific language features improves the discrimination ability of a classifier to detect cyberbullying. In addition to gender, other features were also used, such as profane words, second person pronouns, and the weight of the words obtained with the TF-IDF.

Due to the fact that unlabelled data in online content has grown exponentially, Nahar et al. [10] proposed a session-based one-class classification scheme for automatic detection of cyberbullying by sorting messages based on the time and generated streaming sessions of varying length. Moreover, they incorporated an ensemble of one-class classifiers in this session-based framework, and used swear words and personal pronouns as features, mentioning that the texts with personal pronouns close to a swear word are more likely to have offensive content. TF-IDF was used to represent data by providing appropriate weight to each unigram feature. All the words were converted into lowercase letters, the stop words were removed and the authors also indicated that the proposed approach is reasonably effective and that ensemble learner outperforms the single window and fixed window approaches. An application to detect cyberbullying content on Twitter was developed by Saravanaraj et al. [23], using the network activity, user and tweet contents as features; feature extraction to detect name, gender and age of the bully; and, Naive Bayes and Random Forest as classifiers. Di Capua et al. [24] proposed an unsupervised approach based on Feature Engineering and a model inspired by Growing Hierarchical Self-Organizing Maps [25], capable to cluster documents containing cyberbullying content.

This model was fine-tuned with Twitter texts and tested with YouTube and Formspring texts. The features used in this research were syntactic, semantic, sentiment, and social.

An approach to detect cyberbullying from posts written by bullies and victims from ASK.fm corpus for English and Dutch, using SVM as classifier was presented by Van Hee et al. [26], proving that word N-grams, character N-grams and subjectivity lexicons are strong features for this task. Tokenization, Part-of-Speech (POS)-tagging and Lemmatization were applied as pre-processing. After the optimisation of the hyperparameters, an F1-score of 0.64 and 0.61 was achieved for English and Dutch, respectively. Rosa et al. [27] conducted a review of 22 studies on automatic cyberbullying detection and tested various scenarios with two datasets to validate the practices used in this task, using several techniques such as TF-IDF, textual features, sentiment features, Word Embeddings, personality trait features, and Medical Research Council features. SVM, Logistic Regression, and Random Forest classifiers were used as classifiers. The scenario with the best results was TF-IDF, with personality traits features and Word Embeddings. According to the authors, the current practice of performing Feature Engineering to improve classification performance is, at best, marginally better and the Fβ-score is the best evaluation metric for this classification problem, especially because it is possible to parameterize β to become F1-score or F2-score.

Hani et al. [28] realized several experiments with the Formspring dataset, proposing a supervised Machine Learning approach for detecting and preventing cyberbullying with SVM and Neural Network (NN) as classifiers. The performance of both classifiers was compared on both TF-IDF and sentiment analysis feature extraction methods. Tokenization, lowercase transformation, stop words removal, encoding cleaning, and spell correction were applied as pre-processing. TF-IDF was used to extract the features of the input data, the sentiment analysis technique was used to extract the polarity of the sentences, the approach N-Gram to consider the different combinations of the words during evaluation of the model, namely 2-gram, 3-gram, and 4-gram. To remedy the data imbalance, they took the same number instances of both classes to measure Accuracy. Very large texts and noisy data have been removed. The results show that NN performed better, achieving an F1-score of 0.919, while SVM achieved an F1-score of 0.898. Sanchez and Kumar [29] resorted to sentiment analysis to detect offensive texts in Twitter using Naive Bayes as classifier and boolean word feature extraction. All facts that did not express opinions like news and objective phrases were removed, the keywords were extracted from the text and BoW was used, where every word was a feature with a value of True. A sentiment classifier was developed by Nalini and Jaba Sheela [5], using Latent Dirichlet Allocation (LDA) with Naive Bayes on a Twitter corpus with the main purpose of using sentiment analysis to detect offensive texts. LDA was used to identified the n key terms and the best results were achieved when it reached 2000 key terms with an Precision of 0.705, Recall of 0.706 and F1-score of 0.704. Bigelow et al. [30] developed a search system using Latent Semantic Indexing (LSI) for the detection of cyberbullying

on the Formspring dataset. In this system, at least two workers had to identify the post as offensive for it to be recorded as a positive instance. The common emoticons were replaced by texts equivalents, the common abbreviations were converted in the correct words, the misspellings were corrected, all words were converted to lowercase and all texts with more than 1000 words or less than 3 words were removed. The authors pointed out that this system is not dependent on a dictionary of bullying terms, and, therefore, relies on LSI for semantic analysis.

A Fuzzy rule-based system with Genetic Algorithms to detect offensive texts in Myspace and Formspring was proposed by Nandhini and Sheeba [31]. The evolutionary process was done using a Genetic Algorithm, the evaluation of the chromosome was done using a Fuzzy rule set, and features such as noun, adjective, pronoun and statistics on occurrence of word in the text were extracted. As pre-processing step, stop words, extra characters, and hyperlinks, among others, were removed. Rosa et al. [32] built a Fuzzy Fingerprints model to detect cyberbullying, based on the weights of the top-n most frequent words from the Formspring dataset, achieving an F1-score of 0.425 for the class that contains offensive content. They mentioned that results related to the class with offensive content are more important that the class without this same content and, that it is more important to have a good Recall than a good Precision.

Raisi and Huang [33] proposed a model that discovers bullies and victims as well as new bullying vocabulary, using an objective function based on participant-vocabulary consistency. Twitter and ASK.fm were used as datasets. A system to detect cyberbullying content was proposed by Sherly and Jeetha [34], using Supervised Feature Selection by a Ranking method in order to choose the features and Extreme Learning Machine as classifier. The authors showed that incorporating the Supervised Feature Selection with Extreme Learning Machine produced better results than Extreme Learning Machine alone.

Noviantho et al. [35] performed experiments using the Formspring dataset, but instead of the two classes, they used eleven classes with severity degrees. Texts that had less than 15 letters were eliminated. Tokenization, lowercase transformation, stop words removal, Stemming, and N-grams generation were applied as pre-processing. Naive Bayes and SVM with Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid kernels were used as classifiers. In these circumstances, the most optimal SVM kernel was the Polynomial kernel with an average Accuracy of 0.971.

3.2. Deep Learning Approaches

Social media is, in general, considered a noisy information source. Zhang et al. [36] proposes a pronunciation-based CNN to mitigate the noise problem and sparsity of offensive data, by using phoneme codes of the text as features to correct the spelling errors that did not change the pronunciation. Their experiments on the Formspring dataset achieved an F1-score of 0.571 and their experiments on a dataset of tweets achieved an F1-score of 0.983. A robust text representation is a critical issue in the automatic detection of bullying messages. So, Conneau et al. [37] presented a deep CNN for text processing which

operates directly at the character level, using up to 29 convolutional layers, concluding that max-pooling performs better than other pooling types and that a deeper network achieved an improved performance. Zhao and Mao [38] proposed a representation learning method to tackle this problem that is able to exploit the hidden feature structure of bullying information, and learn a robust and discriminative representation of text. Word Embeddings were used to automatically expand and refine offensive word lists, initialized by domain knowledge. Their experiments on two public cyberbullying data reveal that their approach outperform other baseline text representation learning methods.

Huang et al. [39] described a real-time cyberbullying intervention interface that makes use of CNN for text classification. They showed that cyberbullying can be identified before it takes place, and provide early feedback about how other people would feel before the message is sent out. This interface gives a chance for the user to revise the text, and provides a system-level warning in a situation related to cyberbullying. In the same vein, Al-Ajlan and Ykhlef [40] proposed an algorithm based on CNN, adapting the concept of Word Embeddings, which incorporate semantics not just features extracted from raw text, eliminating the need for Feature Engineering. Experiments showed that this approach outperforms classic cyberbullying detection approaches, achieving an Accuracy of 0.95. An empirical study was conducted by Al-Hashedi et al. [9] to evaluate the robustness of DNN in cyberbullying detection, using the Formspring dataset. Three architectures were experimented: Gated Recurrent Unit (GRU), LSTM and BiLSTM. Four different Word Embeddings were explored, including Word2vec, GloVe, Reddit and Embeddings from Language Models (ELMo). Adaptive Synthetic (ADASYN) [41] was used to generate more synthetic data, helping to deal with imbalance problem, which according to the authors, it is a much better technique than Class Weights. To prevent overfitting, the value of 0.2 in dropout and recurrent dropout were used as parameters of the layer. The results showed that BiLSTM using ELMo outperformed all other models. Based on current research about cyberbullying detection, Rosa et al. [42] implemented three architectures to try solve this problem, which are CNN, CNN-LSTM, and CNN-LSTM-DNN. In addition, they used the Formspring dataset and three text representations trained using Google-News, Twitter, and Formspring. The experimental results showed that these models outperformed other classical classifiers, such as SVM and Logistic Regression, with the best result being an F1-score of 0.444 for the class that contains offensive content. A broad work related to the detection of cyberbullying was carried out by Dadvar and Eckert [43], which involved the reproduction and validation of findings in literature with the same datasets used by the authors, namely Formspring, Wikipedia, Twitter and YouTube. They evaluated the performance of CNN, LSTM, BiLSTM, and BiLSTM with Attention as well as several Transfer Learning approaches. The best results were achieved with BiLSTM with Attention and Model Level Transfer Learning. They also mentioned that models like these can benefit from the integration of other sources of information. Agrawal and Awekar [44] also performed extensive experiments using Transfer Learning

with the aim of detecting cyberbullying on three well-known datasets, which are Formspring, Twitter, and Wikipedia. A set of DNN was used, namely CNN, LSTM, BiLSTM, and BiLSTM with Attention. It was concluded that these models can be applied to one dataset and be transferred to another.

Hate speech, closely related with cyberbullying and offensive speech, can be represented in many ways, making it very difficult identify the degree of aggression and intention. Deep Learning approaches can be used to solve problems like this, therefore Kapil' et al. [45] proposed the use of CNN, LSTM, BiLSTM, and Character-CNN, using various Word Embeddings, namely Word2vec, GloVe, and fastText for detecting several types of hate speech. The best Accuracy and F1-score were obtained at 5 epochs and batch size of 32, and the best performing models were LSTM and BiLSTM. Zhang et al. [46] introduced a combination of CNN and LSTM to detect hate speech, achieving good results in terms of F1-score on six out of the seven datasets used. They also proved that that the model learns better with pre-trained word vectors. Van Huynh et al. [47] implemented a system to solve the problem in the "Vietnamese Language and Speech Processing shared task 2019: Hate Speech Detection on Social Networks", testing three different models, such as CNN, Bidirectional Gated Recurrent Unit (BiGRU)-CNN, and BiGRU-BiLSTM-CNN. Regarding the Word Embeddings, they used fastText. Results of this task showed that BiGRU-BiLSTM-CNN achieved the best performance among these models with an F1-score of 0.705. Another work that studied Deep Learning approaches to detect hate speech was done by Mozafari et al. [48], who used two datasets that were annotated for racism, sexism, hate, and offensive content on Twitter and experimented different combinations of Bidirectional Encoder Representations from Transformers (BERT) with other models, such as CNN and LSTM. The evaluation results indicated that BERT-CNN outperformed previous works by profiting from the syntactical and contextual information embedded in different transformer encoder layers of the BERT using a CNN fine-tuning strategy.

Concerning the identification of offensive language, a inherently related task to cyberbullying and hate speech, Ong [49] experimented a set of DNN to understand which architecture is best suited for this detection. This set of models ranged from CNN, LSTM, BiLSTM, GRU, BiGRU, and combinations among these models. They have adopted GloVe word vectors, some pre-trained using Twitter with 100 and 200 dimensions, and others pre-trained with Common Crawl with 300 dimensions. Synthetic Minority Oversampling TEchnique (SMOTE) and Class Weights were used to counter the imbalance between classes and the author found that by feeding data into the LSTM layer first, then followed by CNN layer produced much better results than the alternative, concluding that the architecture that gave the highest macro average F1-score was BiLSTM-CNN.

The identification of offensive language has recently been the focus of OffensEval [50], a series of shared tasks on offensive language identification, conducted in the scope of SemEVAL [51, 52], the well-known International Workshop on Semantic Evaluation. The

tasks that have occurred so far have been the "Identifying and Categorizing Offensive Language in Social Media (OffensEval 2019)" [53] with the Offensive Language Identification Dataset (OLID) [54] and the "Multilingual Offensive Language Identification in Social Media (OffensEval 2020)" [55] with the Semi-Supervised Offensive Language Identification Dataset (SOLID) [56]. In OffensEval 2019, Zampieri et al. [53] mentioned that among the top-10 teams, seven used BERT with variations in the parameters, Liu et al. [57] achieved the best result with a macro average F1-score of 0.829, using pre-trained BERT with fine-tuning on the OLID, and hashtag segmentation and emoji substitution as pre-processing. In OffensEval 2020, Zampieri et al. [55] emphasized that many teams used context-independent embeddings from Word2vec or GloVe and some works resorted to other Transfer Learning approaches [43, 44] or Multi-Task Learning (MTL) [58]. In terms of models, the ones used were BERT [59], Multilingual BERT (mBERT) [59], Robustly Optimized BERT Pretraining Approach (RoBERTa) [60], Cross-Lingual Language Robustly Optimized BERT Pretraining Approach (XLM-RoBERTa) [61], A Lite Bidirectional Encoder Representations from Transformers (ALBERT) [62] and Generative Pre-trained Transformer 2 (GPT-2) [63]. Most of the teams performed some of the following pre-processing: conversion of emojis into word representations, hashtag segmentation, abbreviation expansion, bad word replacement, spell correction, lowercase conversion, Stemming, and Lemmatization. Other techniques included the removal of user mentions, URLs, hashtags, emojis, e-mails, dates, numbers, punctuation, consecutive character repetitions, offensive words, and stop words. The best result was a macro average F1-score of 0.92, using an ensemble of ALBERT models of different sizes and the OLID to train.

3.3. Summary of Literature Review

Classical approaches are the most common approaches used to detect offensive texts on social media. Proper Feature Engineering, good word representation, refined pre-processing and the use of classical Machine Learning algorithms are the main characteristics to be taken into account in these approaches. Feature Engineering is the most relevant approach among the classical approaches and is still widely used in this area. Table D1 presents a number of works concerning the use of classical approaches, summarily describing the pre-processing tasks, techniques, models, datasets, and achieved performance.

More recently, classical approaches have increasingly been replaced by Deep Learning approaches due to their tendency to achieve better performance. These new approaches, based in DNN, revolutionised the paradigm around text classification. Experiments using Deep Learning methods started to be carried out with the implementation of several DNN architectures, namely CNN, and LSTM, among others. Meanwhile, Word Embeddings emerged, such as Word2vec, GloVe, and fastText, allowing to create pre-trained word embeddings models which are embeddings trained on large datasets for specific tasks that can be used for other tasks. These word representations have created a new form of learning that can provide an increase in terms of performance. Although DNN using

these pre-trained word embeddings has achieved good results, new approaches like Attention mechanism emerged creating new models, called Transformers. BERT was the first Transformer-based Machine Learning technique, serving as the basis for others, such as mBERT, RoBERTa, XLM-RoBERTa, ALBERT, and GPT-2, among others. At the moment, these models are the ones that allow achieving state-of-the-art results. Table D2 presents a set of works concerning the use of Deep Learning approaches, briefly describing the pre-processing tasks, techniques, models, datasets, and achieved performance.

CHAPTER 4

Data

This chapter is devoted to the presentation and description of the datasets, as well as to the data pre-processing for the main approaches covered in this work: classical Machine Learning approach and Deep Learning approach. Section 4.1 describes the datasets used in this work, and Section 4.2 presents the data preparation steps.

4.1. Datasets

Although these problems are often discussed, not much data is available because labeling these texts is a very difficult task. Few datasets are publicly available and these are considered relatively small. This section is devoted to the presentation and description of the three datasets used in the scope of this work.

4.1.1. Formspring Dataset

Formspring was a question-and-answer-based social network founded in 2009, where users asked and answered questions. Among several options, the website had an anonymity option, which increased the occurrence of cyberbullying. In 2015, the website was closed and, to fill that void, similar websites like AskFM and Tumblr took their place.

A version of the Formspring dataset collected by Reynolds et al. [64] was used in this work. The dataset contains the profile information of 50 Formspring users, and the corresponding posts, where each post was labeled with the presence of cyberbullying content by three workers of the Amazon's Mechanical Turk. This work uses only the post text and the corresponding label, where each post contains a question, and the corresponding answer. Example:

> Q: what's your favorite song? :D
> A: I like too many songs to have a favorite.

The dataset has 12852 labeled texts, 1038 of them labeled as having cyberbullying, by at least two workers, and 11814 as having no such content. In order to create the training and testing sets, I have shuffled the data and selected 80% for training and the remaining 20% for testing. Table 4.1 shows the distribution of classes for these sets. Each text contains 1 to 1083 words, and 23.38 is the average number of words. Figure 4.1 shows the most frequent words, both in the negative and in the positive classes.

This dataset is described as being a cyberbullying dataset, but its content should be considered as offensive language instead. That is because cyberbullying implies an offensive action to be carried out over time, by the same person or group [2]. In this case, the texts were labeled as isolated offenses.

FIGURE 4.1. The most frequent words of the negative class (left) and of the positive class (right) in the Formspring dataset

4.1.2. OLID

OLID [54] is a collection of English texts that was used in the "SemEval 2019 challenge: OffensEval 2019 – Identifying and Categorizing Offensive Language in Social Media (Task 6)". It was annotated following an hierarchical three-level annotation model, having the following three sub-tasks, corresponding to the three levels:

- Sub-task A: Offensive language identification;
- Sub-task B: Automatic categorization of offense types;
- Sub-task C: Offense target identification.

Sub-tasks B and C were not considered in this work because these tasks has as their main focus the detection of the type and target of offenses, respectively. Regarding the sub-task A, the main goal is differentiate between offensive and non offensive texts. Insults, threats and texts containing profane language or swear words are often present in offensive texts.

The data source is the Twitter, a social network created in 2006, where users can interact with each other by publishing short texts known as tweets. Registered users publish tweets up to 280 characters, while unregistered users can only read them.

This dataset contains two subsets: the training set and testing set, and their distributions are shown in Table 4.1. For anonymization purposes, all user mentions were replaced by '@USER' in all texts. The average number of words is 19.7 words, with each text containing from 1 to 62 words. The most frequent words present in the negative class and in the positive class are shown in Figure 4.2.

FIGURE 4.2. The most frequent words of the negative class (left) and of the positive class (right) in the OLID

4.1.3. SOLID

SOLID [56] is a dataset used in the "SemEval 2020 challenge: OffensEval 2020 – Multilingual Offensive Language Identification in Social Media (Task 12)". It follows the annotation schema used in OffensEval 2019, so sub-tasks B and C were again not considered. It covers five languages: Arabic, Danish, English, Greek, and Turkish. In this work only the English dataset was used. The training set contains 9089140 English tweets, labeled in a semi-supervised manner using democratic co-training, with the OLID as a seed dataset.

Since the SOLID training set was not annotated manually, I used the OLID training set for training, and the SOLID testing set for testing. Each text of testing set contains from 2 to 42 words, and the average number of words is 13.94. Table 4.1 shows the distribution of classes for the testing set. Figure 4.3 shows the most frequent words present in the negative class and in the positive class.

FIGURE 4.3. The most frequent words of the negative class (left) and of the positive class (right) in the SOLID

TABLE 4.1. Number of instances in the training and testing sets

Dataset	Training		Testing	
	Negative Class	Positive Class	Negative Class	Positive Class
Formspring	9484	805	2330	233
OLID	8840	4400	620	240
SOLID	8840	4400	2907	1080

4.2. Data Preparation

Data pre-processing is described in this section, first for the classical Machine Learning approach and subsequently for the Deep Learning approach. These tasks allow transforming the data that is subsequently used by the models, thus creating better conditions for the models to achieve better performance.

4.2.1. Pre-Processing for the Classical Machine Learning Approach

The text cleaning and pre-processing are important tasks in this type of work and when properly applied allow to achieve better results. Firstly, several external files were imported to assist both tasks with help of specific websites, namely:

- A list of HTML tags and encoding representations [65]. Example: ''';
- An emoticon dictionary created in the context of this work, with the emoticon code associated with an expression. Example: ':) ⇒ smile';
- A list of bad words without duplicates, composed by the aggregation of several files [66–70];
- A slang dictionary, with the slang and the respectively correct expression [71]. Example: 'muck ⇒ ugly').

Prior to pre-processing, the text cleaning was applied to the datasets as follows:

- Encoding representations removal with the help of imported encoding list;
- Unicode characters removal;
- HTML tag conversion;
- 'Q:' and 'A:' expressions removal (only in Formspring dataset).

The pre-processing is crucial for this process and involves several steps. In general, these tasks consist of transforming and removing the noisy content from the text. The order of these steps is important because the previous step may have an impact on the next step. These steps are described in the order in which they were implemented:

- Lowercase conversion;
- URL and username mentions removal;
- Contractions treatment. Example: 'what's' becomes 'what is';
- Elongated words treatment, convert words with a sequence of characters that are repeated more than twice to a maximum of two equal characters. Example: 'cooooool' becomes 'cool';
- Regularize certain expressions. Example: 'muahahah' becomes 'muah';
- Convert identical words together into the same separate words. Example: 'uglyuglyugly' becomes 'ugly ugly ugly';
- Convert emoticons into the associated expression, based on the emoticons dictionary;
- Convert slang into the associated expression, based on the slang dictionary;
- Removal of punctuation, symbols and words with punctuation or symbol, except bad words from the imported bad words list;
- Isolated number removal;
- Spell correction using Spello [72];
- Get the profanity for each text by applying Profanity-Check [73];
- Get the sentiment polarity for each text using Valence Aware Dictionary and sEntiment Reasoner (VADER) [74];
- Stop words removal;
- Removal of isolated characters.

Here follows a summary of the applied steps: lowercase conversion, contraction treatment, elongated words treatment, regularize certain expressions, convert identical words together, convert emotions, convert slang, removal of punctuation and spell correction.

Another important aspect is removing expressions such as URL mentions, username mentions, isolated number, multiple white space, stop words and one character words, which can have a negative impact on the creation of features and as well as on final inputs. Finally, getting the profanity and sentiment of the texts are key steps in creation of profanity features and sentiment features explained ahead in this section. It is important to get the profanity and sentiment of the texts before Feature Engineering because at this stage some words are removed, causing some meaning to be lost from the texts.

There are several common features in the offensive texts of these datasets. The most of these texts have profane words, negative sentiment and are classified as explicit expressions [9], as mentioned in Section 1.1. The social media texts have common features, regardless of being classified as cyberbullying, offensive language, or hate speech. Some of them are the presence of URLs, username mention, elongated words, identical words together, emoticons, emojis, slang, and misspellings, among others. These features may represent some form of offense, depending on several factors such as irony, sarcasm or subjectivity, being these expressions classified as implicit, also as mentioned in Section 1.1.

The proposed approach consists of using the following types of features: grammatical features, bad words features, profanity features, and sentiment features. Grammatical features refer to the most important grammatical classes in offensive language, while the bad words features, profanity features, and sentiment features are features added to the computational representation of the texts. The basis of this representation lies on the concept of word weight, the importance of a feature for the representation of a document. I explored the individual use of these features to better understand their specific contribution, but, as expected, results are better when using them combined.

Grammatical Features

The grammatical classes that have the greatest correlation with texts related to offensive content are nouns, verbs, adjectives, adverbs, and pronouns [75]. Based on this argument, I kept nouns, verbs, adjectives, adverbs, second person pronouns, third person pronouns, and bad words, removing all the other words. Usually, when the bully writes the offensive text, he refers to the victim in the second and third person, so I decided keep second person pronouns and third person pronouns. Additionally, it is important to keep the bad words in order to be able to create the bad words features. This task was implemented using POS tagging by Natural Language ToolKit (NLTK) [76] and bad words list.

Bad Words Features

Bad words correspond to swearing and offensive words commonly not accepted by society. These features were created based on the number of bad words found in each text. The number of bad words was counted for each text, with the help of the bad words list. If the number of bad words is 0, then the feature `badword0` is added. If the number of bad words is greater than 0, then the feature `badword`i will be added, being i the number

of bad words in the text. For example: the text "*friend faggot*", will be represented as `friend faggot badword1`.

Profanity Features

Profanity describes sentences that are offensive and whose use is frowned on by the society. A profanity sentence is not necessarily a sentence that contains bad words, it may have no such words and be considered offensive. The profanity features were created based on the profanity check tool, applied to each text obtained in pre-processing, adding at the end of the text representation the feature `profanity`i, where i is a number between 0 and 10, corresponding to the level of profanity of the text. If i is close to 10, then the text clearly has profanity. Originally, the value of profanity is a decimal number between 0 and 1, so it is rounded to one decimal class and multiplied by 10 to obtain a integer number.

Sentiment Features

Sentiment analysis is a technique used to determine the polarity of the sentiment present in a document and, in general, offensive language expresses negative sentiments. So, the creation of sentiment features is important to improve classification [5, 24, 27–29]. In the pre-processing of each text, the corresponding sentiments were obtained, which consist in polarity scores: neutral, positive, negative, and compound. I used the negative score, which is a decimal number between 0 and 1, where values close to 0 mean that the text has a less negative sentiment and values close to 1 mean that the text has a more negative sentiment. In the same way as it was done in the construction of profanity features, this value is rounded to one decimal class and multiplied by 10 to obtain a integer number, creating and adding a new feature `sentiment`i at the end of the text, where i is a number between 0 and 10.

Document Representation Approaches

I explored two different strategies for document representation: one is based on BoW representation and TF-IDF weighting scheme, which leads to a standard vectorial representation of each document; the other, based on the number of terms in each document for each class, leading to each document being represented by a single weight, which can be considered its offensiveness level. Figure 4.4 shows these two strategies applied to the document "*friend faggot badword1 profanity7 sentiment7*".

To create the single weight document representation, it is necessary to count the occurrence of each distinct term in each document for each class, i.e., to count the number of times a distinct term appears in documents of a certain class. The next step is to create a weight associated with each term, as defined in Equation 4.1, where N_t represents the number of occurrences of the term t in the documents of a given class.

$$w_t = \frac{N_{t_{positive\ class}}}{N_{t_{positive\ class}} + N_{t_{negative\ class}}} \tag{4.1}$$

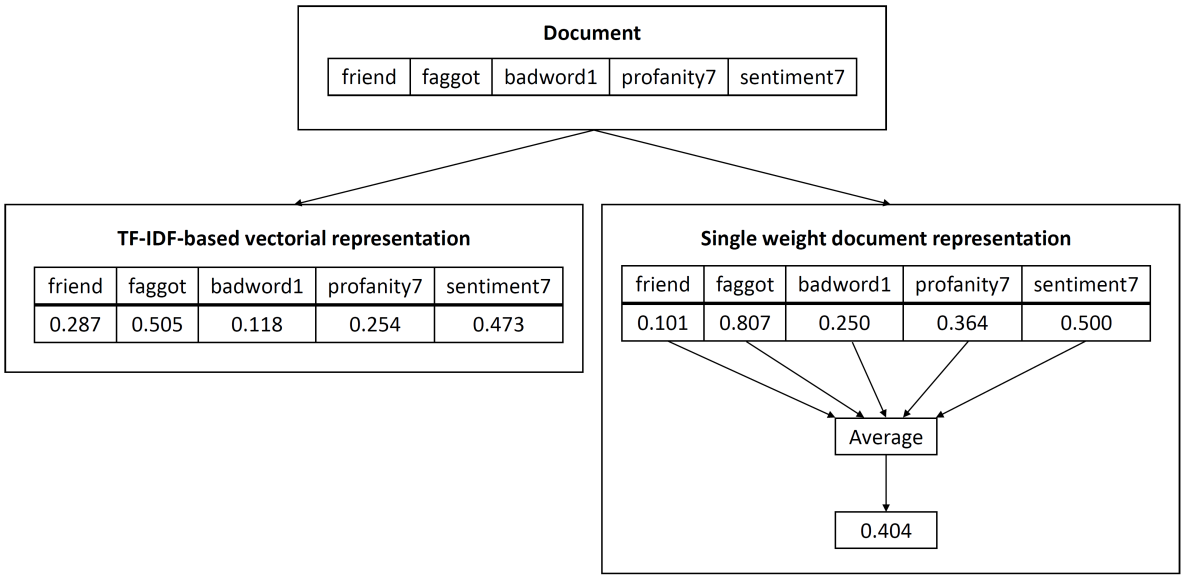


FIGURE 4.4. Strategies for document representation

The weights defined by Equation 4.1 vary between 0 and 1. The closer the weight is to 1, the greater the number of times that term appears in the documents of the positive class and, consequently, the fewer times it appears in the documents of the negative class. The closer the weight is to 0, the greater the number of times that term appears in the documents of negative class and, consequently, the fewer times it appears in the documents of the positive class. Weights with values equals to 0.5 mean that the number of times that term appears in the documents of the positive class is equal to the number of times that term appears in the documents of the negative class. Nevertheless, there are terms that only appear in the documents of the negative class and terms that only appear in the documents of the positive class, leading to weights of 0 and 1, respectively. This means that these terms may have little representation due to the small size of the datasets, appearing only in documents of one class by randomness, or it may mean that these terms may be of great importance if they are representative of offensive language. To minimize this problem, several conditions were implemented to modify these weights. New weights were computed using the function $f()$ as defined in Equation 4.2, which takes a weight w_t as parameter and returns a new weight if w_t is 0 or if w_t is 1, returning w_t, otherwise. It uses the smallest weight, *smallest* w_t, of the set of terms with weight different from 0, the biggest weight, *biggest* w_t, of the set of terms with weight different from 1, the number of occurrences of term t in the documents of a given class ($N_{t_{positive\ class}}$ or $N_{t_{negative\ class}}$) and the total number of documents of the same class ($T_{positive\ class}$ or $T_{negative\ class}$).

$$f(w_t) = \begin{cases} smallest\, w_t - \left(\frac{N_{t_{negative\ class}}}{T_{negative\ class}}\right), & \text{if } w_t = 0 \\ biggest\, w_t - \left(\frac{N_{t_{positive\ class}}}{T_{positive\ class}}\right), & \text{if } w_t = 1 \\ w_t, & \text{otherwise} \end{cases} \tag{4.2}$$

If the new weight resulting from Equation 4.2 is less than 0, then the weight is converted into 0. If the weight is greater than 1, then the weight is converted into 1. The term weights for the document "*friend faggot badword1 profanity7 sentiment7*" is shown in Table 4.2.

TABLE 4.2. An example of the term weights for a document

Word	friend	faggot	badword1	profanity7	sentiment7
Weight	0.101	0.807	0.250	0.364	0.500

At this point all terms have associated weights, so the next step is to create a representation for each document by averaging their term weights, resulting in a value between 0 and 1. The closer the value is to 1, the higher the probability that the document is associated with offensive language. The weight associated to the document "*friend faggot badword1 profanity7 sentiment7*" is $(0.101 + 0.807 + 0.25 + 0.364 + 0.5)/5 = 0.404$.

In general, the document weights are relatively low, since the datasets are imbalanced, with many more documents of the negative class than the positive class, and consequently many more terms with low weight than terms with high weight, so some documents of the positive class have low weights, i.e., these documents have more terms with low weights than terms with high weights.

The representation of documents in the testing set is computed with the same method of each methodology (TF-IDF-based vectorial representation and single weight document representation), using the terms from the training set and their respective weights from each methodology. However, there are terms in the testing set that do not match to any term in the training set and, in this case, it is necessary to find the most similar term in the training set. Table 4.3 shows the total number of terms of the testing set and the number of terms in the testing set that do not match to any term from the training set.

TABLE 4.3. Number of unknown terms in testing set

Dataset	**Total Terms**	**Unknown Terms**
Formspring	37243	1365
OLID	13610	1328
SOLID	39934	2564

The number of unknown terms is small, but it is important to find the most similar terms in the training set because some of these unknown terms may be strong indicators of offensive language, e.g., they may be bad words. To address this issue, the 'get_close_matches' method from the library difflib [77] was used, which returns the most similar term to a given term, being also possible to obtain the list of the n most similar terms. However, there are terms that are not found, so in the end and only in this case, the library fuzzywuzzy [78] was used. This toolkit uses Levenshtein Distance [79]. For the terms in the testing set that match the terms in the training set, the weight associated with the term computed in the training set was used.

In order to improve the results of the single weight document representation, an optimization of the document weights was performed. Firstly, I found the minimum weight and the maximum weight of the distribution of document weights for each class in the training set. Table 4.4 describes the minimum and maximum weights for each class.

TABLE 4.4. Minimum and maximum weights for each class

Dataset	**Class**	**Minimum Weight**	**Maximum Weight**
Formspring	Negative class	0.019	0.374
	Positive class	0.043	0.615
OLID	Negative class	0.109	0.705
	Positive class	0.190	0.828

Secondly, I applied function g, defined in Equation 4.3, using each document weight w_d of the training set and testing set as parameter, as well as the minimum and maximum document weights of both classes, where min means minimum and max means maximum, computed in the training set only. Function g returns the minimum document weight of the negative class if w_d is less than the minimum document weight of the positive class and the minimum document weight of the negative class is less than the minimum document weight of the positive class or it returns the maximum document weight of the positive class if w_d is greater than the maximum document weight of the negative class and the maximum document weight of the negative class is less than the maximum weight document of the positive class, returning w_d, otherwise.

$$g(w_d) = \begin{cases} w_{d_{min_{negative\ class}}}, & \text{if } w_d < w_{d_{min_{positive\ class}}} \wedge \\ & w_{d_{min_{negative\ class}}} < w_{d_{min_{positive\ class}}} \\ w_{d_{max_{positive\ class}}}, & \text{if } w_d > w_{d_{max_{negative\ class}}} \wedge \\ & w_{d_{max_{negative\ class}}} < w_{d_{max_{positive\ class}}} \\ w_d, & \text{otherwise} \end{cases} \tag{4.3}$$

4.2.2. Pre-Processing for the Deep Learning Approach

All experiments, except the ones using RoBERTa, have been performed with two different pre-trained GloVe word vectors [20]:

- GloVe Twitter: 200 dimension vectors, trained using 2 billion tweets, 27 billion tokens, 1.2 million unique words, and uncased;
- GloVe Common Crawl: 300 dimension vectors, trained on 840 billion tokens, 2.2 million unique words, and cased.

The GloVe Twitter was chosen not only because the OLID and SOLID were built from Twitter data, but also because Forsmpring and Twitter contain social media messages. GloVe Common Crawl was chosen because it is a large web archive, composed of petabytes of data collected since 2011, and a source of many and varied information. Depending

on the pre-trained word vector in use, each word was mapped into a vector of 200 or 300 dimensions. I decided to keep all the words in the sequence, since every piece of information may provide signs of offensive language presence.

The pre-processing steps performed aimed at minimizing the Out-of-Vocabulary (OOV) words, therefore maximizing the coverage of the pre-trained word embeddings. I started by building a vocabulary containing all the unique words contained in the texts of dataset, and their frequencies. Then, I checked which of these words could not be represented by the pre-trained GloVe embeddings, corresponding to the OOV words, and converted them into their corresponding normalized forms. The process is as follows:

Step 1:

- Convert URL mentions to '<url>';
- Convert username mentions to '<user>';
- Convert years in numerical to 'year';
- Convert numbers to their normalized word forms;
- Normalize ordinal numbers. Example: '1st' becomes 'first';
- Convert hours representation to 'hours';
- Delete unknown symbols. Example: 'Œ';
- Isolate the symbols that correspond by separating from the words.

Step 2:

- Elongated words treatment, convert words with a sequence of characters that are repeated more than twice to a maximum of two equal characters. Example: 'cooooool' becomes 'cool';
- Convert identical words together into the same separate words. Example: 'uglyuglyugly' becomes 'ugly ugly ugly';
- Convert contractions. Example: 'what's' becomes 'what is';
- Word segmentation using WordSegment [80]. Example: 'youarefat' becomes 'you are fat';

Step 3:

- Spelling correction using Spello [72].

Step 1 consists of converting and normalizing certain mentions and expressions, such as URL mentions, username mentions, years, numbers, ordinal numbers, and hours representation. In addition, the symbols that did not match were removed and the symbols that did match were separated from the words. In step 2, the elongated words, identical words together and contractions were treated. At the end of this step, word segmentation was also applied, separating the words that for some reason are together. Spello [72] was used in step 3 to correct the OOV words that contained spelling errors by replacing the original words with their corrected form. Step 2 and 3 were repeated until the percentage coverage of the previous process was lower or equal to the current percentage coverage,

until there is no improvement in the percentage coverage. The coverages of the embeddings found in the vocabulary and the number of texts for each dataset and pre-trained GloVe embeddings are shown in the Table 4.5.

TABLE 4.5. Coverage of the pre-trained word embeddings after pre-processing

Dataset	**Twitter**		**Common Crawl**	
	Vocabulary	Texts	Vocabulary	Texts
Formspring	99.30%	99.90%	99.16%	99.93%
OLID	98.80%	99.79%	99.43%	99.91%

These tasks were not performed for RoBERTa models because these models use specific tokenizers that tokenizes a text in words or sub-words, converting to ids through a look-up table, helping with noisy data. In this work when the Formspring dataset and RoBERTa were used, the following pre-processing was applied:

- Elongated words treatment, convert words with a sequence of characters that are repeated more than twice to a maximum of two equal characters.

the following pre-processing was applied to OLID or SOLID when RoBERTa was used:

- Convert URL expressions into the token 'HTTP';
- Convert contractions. Example: 'what's' becomes 'what is';
- Convert hashtags into segmented text, using WordSegment. Example: '#youare-fat' becomes 'you are fat'.

CHAPTER 5

Modelling

The description of the models development is illustrated in this chapter, according to the approaches proposed throughout this work, namely the classical Machine Learning approach in Section 5.1 and the Deep Learning approach in Section 5.2. Classical Machine Learning algorithms were used in the classical Machine Learning approach, while more robust and complex algorithms, like DNN, were used in the Deep Learning approach. The imbalanced classification problem is one of the great obstacles present in the classification predictive modeling and in this chapter, solutions for solving this problem are presented.

5.1. Classical Machine Learning Models

The algorithms used in the classical Machine Learning approach were chosen based on the classical Machine Learning algorithms most used for the automatic detection of offensive texts: SVM [21, 22, 26–28, 35], Logistic Regression [27] and Random Forest [23, 27]. Scikit-learn [81] was the library chosen.

5.1.1. Parameter Optimization

Concerning parameter optimization, a 10-Fold Cross-Validation setting was used in all experiments, with a Randomized Search for each fold to find the best combination of algorithm hyperparameters that is performed in the training set. The choice of Randomized Search instead of Grid Search was based on the fact that Randomized Search is much faster, although performance is slightly worse. The following parameters were eligible for optimization:

SVM:

- C: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1;
- Kernel: Linear, Polynomial, RBF, Sigmoid;
- Degree: 1, 2, 3, 4, 5;
- Gamma: scale, auto;
- Tolerance for stopping criteria: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1.

Logistic Regression:

- C: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1;
- Intercept scaling: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1;
- Tolerance for stopping criteria: 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1;
- Algorithm to use in the optimization problem: Newton-Conjugate Gradient (Newton-CG), Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), LIBLINEAR, Stochastic Average Gradient (SAG), SAGA.

Random Forest:

- Criterion: Gini, Entropy;
- The number of trees in the forest: 100, 300, 500, 700, 900;
- The minimum number of samples required to split an internal node: 2, 3, 4, 5, 6;
- The minimum number of samples required to be at a leaf node: 1, 2, 3, 4, 5;
- The number of features to consider when looking for the best split: square root, logarithm base 2.

5.1.2. Imbalanced Classification Problem

Class imbalanced datasets are very common, especially in classification problems where the class distributions of data are highly imbalanced. The datasets described in this work are imbalanced. To minimize the imbalanced classification problem I used the following approaches: Threshold-Moving [82] and Synthetic Minority Oversampling Technique Edited Nearest Neighbours (SMOTEENN) [83].

Threshold-Moving

Many algorithms predict a probability or a class score. Thus, a simple and straightforward approach to improve the performance of such a classifier on an imbalanced classification problem is to fine-tune the threshold used to map the probabilities to the class labels. Threshold-Moving is a technique used to find an optimal threshold in a predefined set of thresholds, with the advantage that this set can be customised, but with a higher calculation cost. Threshold-Moving using Receiver Operating Characteristic (ROC) curve and Precision-Recall curve are two other possibilities to find the optimal threshold. Both techniques are commonly used in imbalanced classification.

The default threshold is 0.5, which can result in worse performance on classification problems that have a large class imbalance. This decision threshold has scores in the range of 0 or 1 and in a binary classification problem, values below the threshold are assigned to negative class and values above or equal to the threshold are assigned to positive class. The first step is to calculate the probabilities of the positive class using 10-fold cross-validation in the training set. Then, a list of thresholds is created. With these thresholds and the probabilities, the optimal threshold that represents the best score of a specific metric is found. In this case, I used the F1-score and the F2-score. I found the optimal threshold for each metric and applied these thresholds in the prediction of the testing set. With the optimal threshold of F1-score, which is the harmonic mean of Precision and Recall, the desired goal was to achieve the best overall result, whereas with the optimal threshold of F2-score, the aim was to increase Recall, raising the number of True Negatives and decreasing the number of False Negatives. As a consequence of this, Precision decreased. In this type of problems, Recall seems to be more important than Precision [32].

SMOTEENN

Oversampling and undersampling are techniques created to solve the imbalanced data problem. Oversampling involves selecting data from the minority class and adding them to the training set. Undersampling involves selecting data from the majority class to delete from the training set. SMOTE [84], an oversampling technique, generates synthetic samples for the minority class, similar to those that already exist, using K-Nearest Neighbors (KNN). These synthetic samples are generated between a chosen sample and its neighbors. This method can generate noisy samples and this issue can be solved by cleaning the space resulting from oversampling. SMOTEENN solves this problem by applying oversampling using SMOTE followed by undersampling using Edited Nearest Neighbours. SMOTEENN was applied to the training sets only, and Table 5.1 shows the number of instances of each class before the application of SMOTEENN, using the TF-IDF-based vectorial representation and the single weight document representation.

TABLE 5.1. Distribution of instances before and after the application of SMOTEENN

Dataset	**Class**	**Before**	**After**	
			TF-IDF	Single Weight Per Document
Formspring	Negative class	9484	2394	6653
	Positive class	805	9341	6821
OLID	Negative class	8840	1344	5985
	Positive class	4400	7853	6210

5.2. Deep Neural Networks

As seen in Section 3.2, CNN, Recurrent Neural Network (RNN) and Transformers are widely used in the detection of offensive language. On this basis, I have developed four models that consider static pre-trained models: CNN, CNN-Attention BiLSTM, and BiLSTM-Attention. In addition to these models, RoBERTa-base was also used to try to achieve even better results, which consists of contextual word embeddings. The models and their respective parameters are described below.

5.2.1. Embeddings Layer

The first layer on CNN, CNN-Attention, BiLSTM, and BiLSTM-Attention is an Embeddings layer, which is an essential layer for neural networks when using text data. Word Embeddings are a dense representation of words in the text. These words are mapped to vectors of real numbers, being a learned representation for text where words that have the same meaning have a similar representation. The Embeddings layer can be defined as a matrix multiplication that transforms words into their corresponding word embeddings and compresses the input feature space into a smaller one.

Before this layer receives the input data, it is necessary to encode the words in integers and pad the sequences to have the same length as the maximum length, being this maximum length defined as the length of the sequence that has the largest number of values. The sequences that were shorter than the sequence with the maximum length were filled with zeros.

As already mentioned, two pre-trained word vectors were used, being this step defined as the creation of a matrix of weights provided by each pre-trained word vector. To create this matrix of weights, all unique words were listed from the training dataset and the embedding weight vector from pre-trained word vector corresponding to each unique word was inserted into the matrix, resulting in a matrix of weights only for words present during training. With this, the Embeddings layer can be seeded with the respective word embedding weights. The parameters of this layer depend on the pre-trained word vectors. Table 5.2 shows the parameters used depending on the pre-trained word vectors, where input is the size of the vocabulary, input length is the length of the input sequences, and dimension is the dimension of the dense embedding.

TABLE 5.2. The parameters of the Embeddings layer

GloVe	**Input**	**Input Length**	**Dimension**
Twitter	14326	1181	200
Common Crawl	14681	1169	300

The Embeddings layer passes the output to an 1D spatial dropout with a rate of 0.2, in order to regularize the learning and to avoid overfitting. Dropout is a commonly used technique to deal with overfitting by ignoring randomly selected neurons during training [85]. Ong [49] suggests the use of 1D spatial dropout instead of the normal dropout.

5.2.2. CNN

The model implemented was adapted from the architecture proposed by Kim [86]. After Embeddings layer and 1D spatial dropout, CNN start with 1D Convolutional layer consisting of 64 filters, window size of 3, padding the input keeping the output with the same length and the Rectified Linear Unit (ReLU) activation function. After 1D Convolutional layer, a 1D Max Pooling layer with the default parameters is added, reducing the dimensionality without losing important features or patterns. The same block of layers (1D Convolutional layer and 1D Max Pooling layer) is added twice more, the second Convolutional layer with the window size of 5 and the third Convolutional layer with the window size of 7. After the third block of layers a Flatten layer was introduced, which is used to flatten the input, converting the matrix to a single array and by a Fully Connected layer or better known as Dense layer with 64 units, ReLU activation function and dropout with a rate of 0.2. Finally, a last Dense layer with only one unit, Sigmoid activation function and initializer for the bias vector with natural logarithm of number of texts labelled as

positive class divided by the number of texts labelled as negative class as the bias, which generates tensors with constant values. This Dense layer performs classification based on the features extracted by the previous layers. These parameters were chosen for the last layer because this type of problem is a classic binary classification problem. This CNN architecture can be seen in Figure E1, as well as the following architecture:

1) Embeddings layer;
2) 1D spatial dropout (rate = 0.2);
3) 1D Convolutional layer (filters = 64, window size = 3, padding = 'same', activation function = 'ReLU');
4) 1D Max Pooling layer;
5) 1D Convolutional layer (filters = 64, window size = 5, padding = 'same', activation function = 'ReLU');
6) 1D Max Pooling layer;
7) 1D Convolutional layer (filters = 64, window size = 7, padding = 'same', activation function = 'ReLU');
8) 1D Max Pooling layer;
9) Flatten layer;
10) Dense layer (units = 64, activation function = 'ReLU');
11) Dropout (rate = 0.2);
12) Dense layer (units = 1, activation function = 'sigmoid', bias initializer = $\log(\frac{\text{number of texts labelled as positive class}}{\text{number of texts labelled as negative class}})$.

5.2.3. CNN-Attention

The Attention layer has proven to be an excellent way to improve results, especially by applying in LSTM or BiLSTM. Although it is not commonly used in CNN, I tried to use the same CNN described in Section 5.2.2 and replaced the Flatten layer (item 9) with an Attention layer that implements an Attention mechanism, with a context/query vector, for temporal data, and masking support [87]. Figure E2 show the CNN-Attention architecture.

5.2.4. BiLSTM

I built BiLSTM with a BiLSTM layer consisting of 64 units and Sigmoid as recurrent activation function, after the Embeddings layer and 1D spatial dropout. As the last layer I added a Dense layer with only one unit, Sigmoid activation function and initializer for the bias vector as described in Section 5.2.2. The BiLSTM architecture used is shown in Figure E3 and described below:

1) Embeddings layer;
2) 1D spatial dropout (rate = 0.2);
3) BiLSTM layer (units = 64, recurrent activation function = 'sigmoid');
4) Dense layer (units = 1, activation function = 'sigmoid', bias initializer = $\log(\frac{\text{number of texts labelled as positive class}}{\text{number of texts labelled as negative class}})$.

5.2.5. BiLSTM-Attention

I have adopted a strategy similar to the one described in Section 5.2.3, where an Attention layer was added between the BiLSTM layer (item 3) and the Dense layer (item 4). An illustration of BiLSTM-Attention architecture is shown in Figure 5.1. The BiLSTM-Attention architecture used in this work is shown in Figure E4.

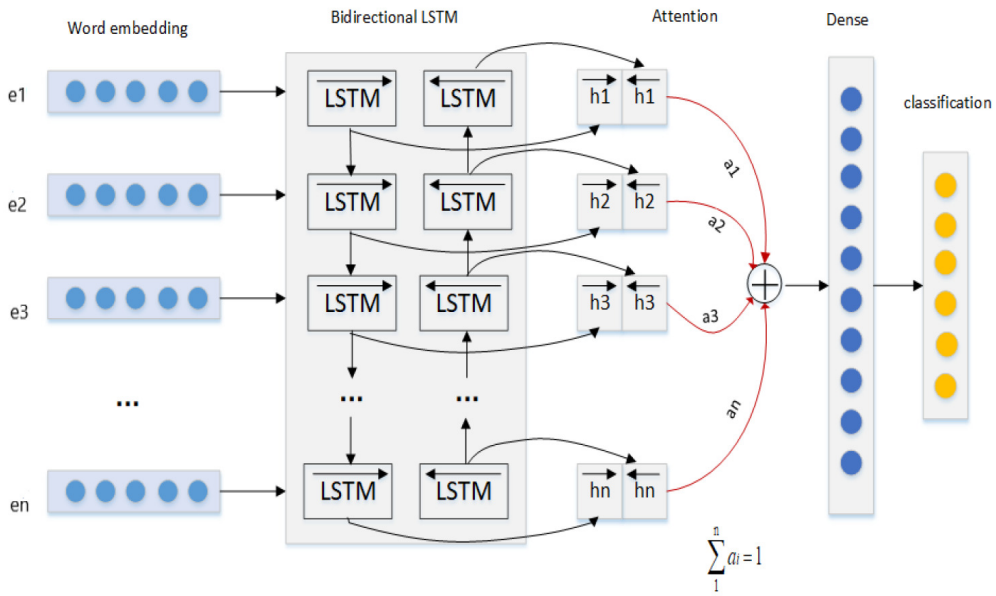


FIGURE 5.1. BiLSTM-Attention architecture [88].

5.2.6. Experimental Setup

All previous models were trained with Adam optimizer and the model stops training if the validation loss does not improve after 10 epochs. The criteria for selecting the best model was always the highest value of validation F1-score. Cyclical Learning Rate (CLR) [89] was used to enhance the way the learning rate is scheduled during training, to provide better convergence, allowing the learning rate to cyclically vary between lower and upper boundary values. The initial learning rate of 1e-7 was the lower boundary in the cycle, upper boundary of 1e-3 in the cycle, number of training iterations per half cycle was of 8 x training iterations in epoch and the mode was exp_range, which is the learning rate that varies between the minimum and maximum boundaries and that at each boundary value declines by an exponential factor.

The approach most often seen in the binary classification problem is to use Accuracy as metric and binary cross-entropy as loss function, but as these datasets are imbalanced, this approach is not the most recommended for this type of datasets. With this approach, the model aims to achieve the highest Accuracy value and this happens by predicting more negative class than positive class. The idea for the proposed approach was to try increase F1-score, which for this case, I think it is more important than Accuracy. For this purpose, F1-score was used as metric and a new Macro Soft-F1 Loss function [90] was implemented. Although a better F1-score is achieved, this approach makes Precision and Recall balanced

between them, and as already referenced, the dataset is extremely imbalanced, so Recall seems to be more important than Precision. For this reason and to solve the imbalanced classification problem, I developed Macro Soft-F2 Loss function based on Macro Soft-F1 Loss function. This loss function allow to increase Recall maintaining a stable F1-score, predicting more positive class, but unfortunately, it failed more negative class.

These models were trained with batch size of 64 and developed using Tensorflow [91] and Keras [92].

5.2.7. RoBERTa

RoBERTa, which is based on Transformers, was introduced by Liu et al. [60] and is similar to BERT. BERT is a multi-layer bidirectional transformer encoder using a combination of masked language modeling goal and next sentence prediction, being trained on the Book Corpus with 800 million tokens and English Wikipedia with 2.5 million tokens. This model was proposed by Vaswani et al. [93], having achieved state-of-the-art results, revolutionizing the whole paradigm that involves NLP until that moment and became the base model for other more recent Transformer models. There are three main types of tokenizers used in Transformers: WordPiece [94], Byte-Pair Encoding (BPE) [95] and SentencePiece [96]. BERT uses WordPiece, while RoBERTa has the same architecture as BERT, but uses BPE and a different pretraining scheme. WordPiece and BPE are very similar techniques that segment words into subwords. In both, a word unit inventory is initialized with all characters of the text and a language model is built on the training data using this inventory. With WordPiece, a new word unit is generated using the combination of two units out of the word unit inventory with the aim of increase the inventory and the new word is chosen from all possible words that maximize the probability on the training data the most when added to the model. With BPE, the new word unit is chosen as the combination of the next most frequently occurring pair among the current set of subword units. Finally, also in both, the process following word unit inventory initialization is repeated until a predefined limit of word units is reached or the probability increase falls below a certain threshold.

In this work, I used the RoBERTa-base model, pre-trained on 58 million tweets, and finetuned for offensive language identification [97] from Hugging Face [98]. This model, given a textual input, returns two outputs, whose sum equals 1. One of these outputs represent the probability that the text is non-offensive and the other output represent the remaining probability that the text is offensive. Only the output representing the offensive probability was used, whereby if an output is greater than or equal to certain threshold it is classified as offensive text and if an output is less than this threshold it is classified as non-offensive text. Macro average F1-score was calculated for each threshold from a list of thresholds between 0 and 1, with a variation of 0.025 between them to find the best threshold for the training set and testing set. The best threshold is the threshold with the highest macro average F1-score and the default threshold is 0.500. The best threshold of training set as well as the default threshold are applied on testing set. The

best threshold of testing set is the threshold that allow to achieve the best results for this subset, representing the maximum capability of this model to correctly predict offensive texts.

CHAPTER 6

Experimental Results

This chapter is devoted to the presentation and analysis of the results from the experiments carried out in this work. All approaches and models explained above are part of these experiments, divided into two main sections: classical Machine Learning approach in Section 6.1 and Deep Learning approach in Section 6.2. The results of the experiments are presented according to macro average and positive class.

The main evaluation metrics reported in the results are Precision, Recall, F1-score and Accuracy. In balanced classification the most important evaluation metric is Accuracy, however, in imbalanced classification, the most relevant evaluation metrics to take into consideration are Recall and F1-score. Rosa et al. [27, 32, 42], Dadvar and Eckert [43], Agrawal and Awekar [44] highlighted that positive class F1-score is the metric most important in this type of problems, while Nahar et al. [10], Di Capua et al. [24], Zhao and Mao [38], Ong [49], Zampieri et al. [53, 55], Liu et al. [57], Dai et al. [58] used macro average F1-score as the main metric to evaluate the performance of the models.

Except for the RoBERTa model, all the others were trained with the training sets and tested with the testing sets. The results of experiments for RoBERTa are shown using the testing set to test the model performance and three thresholds: the best training set threshold, the best testing set threshold and the default threshold. These thresholds were used to understand which is the best method, whether it is finding the best training set threshold and using this in the testing set or whether the results from the default threshold is acceptable enough compared to the best results, which are achieved using the best testing set threshold.

6.1. Experimental Results for the Classical Machine Learning Approach

The results of the models using the classical Machine Learning approach, which includes the TF-IDF-based vectorial representation and the single weight document representation, as well as the methods such as F1-score optimal threshold, F2-score optimal threshold and SMOTEENN are reported in this section. These methods, which were presented earlier, were applied as part of these experiments, allowing to get a better insight into the predictive ability of the models according to the use of each methodology and method.

The results of the models using F1-score optimal threshold as method to improve F1-score for the Formspring dataset can be seen in Tables 6.1 and 6.2, for the OLID in Tables 6.3 and 6.4, and for the SOLID in Tables 6.5 and 6.6. Analysing the F1-score, it should be noted that I achieved state-of-the-art results, surpassing previous work that used classical approaches, and this is due to the fact that the features created

have a significant impact, as well as the use of Threshold-Moving. For the TF-IDF-based vectorial representation, Logistic Regression is the best model in all datasets. In addition, it has the best macro average F1-score and positive class F1-score compared to the single weight document representation. For the Formspring dataset, using the single weight document representation, Random Forest has the best F1-score, while Logistic Regression has the best Recall and SVM has the best Precision and Accuracy. This indicates that Random Forest has the best overall performance, even for the positive class, while Logistic Regression achieves the better Recall in the positive class. On the other hand, SVM has more negative class correct predictions, allowing it to be the model with the best accuracy. The best model for the OLID is Logistic Regression for all evaluation metrics. In relation to the SOLID, SVM has the best macro average F1-score and the same positive class F1-score as Random Forest.

TABLE 6.1. Results for the Formspring dataset using F1-score optimal threshold and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.720	0.712	0.716	0.493	0.472	0.482	0.908
Logistic Regression	**0.752**	0.715	**0.732**	**0.556**	0.468	**0.508**	**0.918**
Random Forest	0.690	**0.732**	0.708	0.427	**0.536**	0.475	0.893

TABLE 6.2. Results for the Formspring dataset using F1-score optimal threshold and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	**0.735**	0.688	0.708	**0.527**	0.412	0.463	**0.913**
Logistic Regression	0.696	**0.725**	0.709	0.441	**0.515**	0.475	0.897
Random Forest	0.702	0.721	**0.711**	0.453	0.502	**0.477**	0.900

TABLE 6.3. Results for the OLID using F1-score optimal threshold and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.700	0.726	0.708	0.540	0.675	0.600	0.749
Logistic Regression	**0.736**	**0.758**	**0.745**	**0.599**	**0.696**	**0.644**	**0.785**
Random Forest	0.717	0.746	0.726	0.561	0.704	0.625	0.764

TABLE 6.4. Results for the OLID using F1-score optimal threshold and single weight document representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.702	0.719	0.708	0.552	0.637	0.592	0.755
Logistic Regression	**0.707**	**0.729**	**0.715**	**0.556**	**0.662**	**0.605**	**0.758**
Random Forest	0.703	0.721	0.710	0.554	0.642	0.595	0.756

TABLE 6.5. Results for the SOLID using F1-score optimal threshold and TF-IDF-based vectorial representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.854	0.912	0.872	0.721	0.969	0.827	0.887
Logistic Regression	**0.881**	**0.934**	**0.899**	**0.770**	**0.980**	**0.862**	**0.913**
Random Forest	0.854	0.910	0.872	0.725	0.961	0.826	0.888

TABLE 6.6. Results for the SOLID using F1-score optimal threshold and single weight document representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	**0.850**	**0.897**	**0.867**	**0.732**	0.925	**0.817**	**0.885**
Logistic Regression	0.847	0.896	0.864	0.726	**0.927**	0.814	0.883
Random Forest	0.849	**0.897**	0.866	0.731	0.925	**0.817**	0.884

Tables 6.7 and 6.8 shows the results of F2-score optimal threshold for the Formspring dataset, Tables 6.9 and 6.10 for the OLID, and Tables 6.11 and 6.12 for the SOLID. Overall, compared to F1-score optimal threshold for the positive class, Recall increased and Precision decreased, as expected. Accuracy decreased for both approaches, with the TF-IDF-based vectorial representation performing worse than the single weight document representation. F1-score increased slightly for some models in the Formspring dataset when the single weight document representation was used. The rationale behind the application of this method was the attempt to give greater weight to Recall than to Precision, predicting more occurrences of positive class. As previously mentioned, it is less problematic that a negative class text is mistakenly considered as positive class than the other way around. Therefore, it is better to predict correctly more instances of the positive class and, in counterpart, miss more instances of the negative class. Logistic Regression is the best model for all datasets in the TF-IDF-based vectorial representation. For the the single weight document representation, SVM has the best Recall and F1-score in the Formspring dataset, while Logistic Regression has the best F1-score for the OLID and SOLID. The choice between F1-score optimal threshold and F2-score optimal threshold

depends on the willingness to sacrifice more negative class instances in order to predict more correctly positive class instances.

TABLE 6.7. Results for the Formspring dataset using F2-score optimal threshold and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.627	0.747	0.652	0.294	0.648	0.405	0.827
Logistic Regression	**0.636**	**0.768**	**0.663**	**0.308**	**0.691**	**0.426**	**0.831**
Random Forest	0.629	0.760	0.653	0.294	0.682	0.411	0.823

TABLE 6.8. Results for the Formspring dataset using F2-score optimal threshold and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.687	**0.758**	**0.714**	0.415	**0.601**	**0.491**	0.887
Logistic Regression	0.696	0.725	0.709	0.441	0.515	0.475	0.897
Random Forest	**0.702**	0.721	0.711	**0.453**	0.502	0.477	**0.900**

TABLE 6.9. Results for the OLID using F2-score optimal threshold and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.627	0.616	0.478	0.340	**0.925**	0.498	0.479
Logistic Regression	**0.651**	**0.671**	0.570	0.385	0.896	**0.539**	0.572
Random Forest	0.642	0.667	**0.580**	**0.389**	0.854	0.535	**0.585**

TABLE 6.10. Results for the OLID using F2-score optimal threshold and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.664	0.704	0.643	0.439	0.808	0.569	0.658
Logistic Regression	**0.680**	**0.722**	**0.674**	**0.473**	0.779	**0.589**	**0.697**
Random Forest	0.642	0.665	0.572	0.384	**0.867**	0.533	0.576

TABLE 6.11. Results for the SOLID using F2-score optimal threshold and TF-IDF-based vectorial representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.711	0.736	0.618	0.422	**1.000**	0.593	0.619
Logistic Regression	**0.777**	**0.845**	**0.765**	**0.555**	0.997	**0.713**	**0.777**
Random Forest	0.750	0.808	0.716	0.502	0.996	0.668	0.725

TABLE 6.12. Results for the SOLID using F2-score optimal threshold and single weight document representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.813	0.881	0.823	0.638	**0.974**	0.771	0.839
Logistic Regression	**0.823**	**0.888**	**0.837**	**0.663**	0.965	**0.786**	**0.854**
Random Forest	0.821	0.886	0.835	0.657	0.967	0.783	0.851

Tables 6.13 and 6.14 shows the results of applying SMOTEENN on the training sets for the Formspring dataset, Tables 6.15 and 6.16 for the OLID, and Tables 6.17 and 6.18 for the SOLID. SMOTEENN made it possible to balance the number of classes and consequently, it allowed to predict the instances of the positive class more correctly, increasing Recall, while Precision worsened. Overall, this approach behaves in a similar way to F2-score optimal threshold, with the TF-IDF-based vectorial representation performing worse than the single weight document representation. SVM is the best model in all datasets of the TF-IDF-based vectorial representation. For the single weight document representation, the best model in the Formspring dataset and SOLID is Logistic Regression, having the best F1-score, while SVM is the model with the best Recall and F1-score in the OLID.

TABLE 6.13. Results for the Formspring dataset using SMOTEENN and TF-IDF-based vectorial representation

Model	Macro Average			Positive Class			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	**0.605**	**0.736**	**0.618**	**0.250**	0.674	**0.365**	**0.787**
Logistic Regression	0.562	0.684	0.445	0.145	**0.893**	0.250	0.514
Random Forest	0.584	0.734	0.559	0.198	0.785	0.316	0.692

TABLE 6.14. Results for the Formspring dataset using SMOTEENN and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.655	**0.759**	0.685	0.350	**0.635**	0.451	0.860
Logistic Regression	**0.676**	0.756	**0.705**	**0.394**	0.605	**0.477**	**0.880**
Random Forest	0.632	0.746	0.658	0.305	**0.635**	0.412	0.836

TABLE 6.15. Results for the OLID using SMOTEENN and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.615	**0.591**	**0.436**	**0.325**	0.933	**0.482**	**0.440**
Logistic Regression	**0.616**	0.531	0.294	0.292	**0.988**	0.451	0.329
Random Forest	0.591	0.550	0.359	0.302	0.946	0.458	0.374

TABLE 6.16. Results for the OLID using SMOTEENN and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	**0.705**	**0.739**	**0.714**	0.536	**0.721**	**0.615**	0.748
Logistic Regression	**0.705**	0.735	0.713	**0.540**	0.700	0.610	**0.750**
Random Forest	0.680	0.709	0.687	0.505	0.675	0.578	0.724

TABLE 6.17. Results for the SOLID using SMOTEENN and TF-IDF-based vectorial representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	**0.703**	**0.720**	**0.597**	**0.408**	0.997	**0.579**	**0.597**
Logistic Regression	0.659	0.587	0.390	0.318	**1.000**	0.483	0.404
Random Forest	0.686	0.685	0.549	0.381	0.992	0.550	0.549

TABLE 6.18. Results for the SOLID using SMOTEENN and single weight document representation

Model	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
SVM	0.838	**0.893**	0.855	0.704	**0.938**	0.804	0.873
Logistic Regression	**0.839**	**0.893**	**0.856**	**0.708**	0.933	**0.805**	**0.874**
Random Forest	0.833	0.888	0.850	0.696	0.933	0.797	0.868

6.2. Experimental Results for the the Deep Learning Approach

The results of the experiments presented in this section, use the proposed Deep Learning models and two variants of the pre-trained word vectors. The training process of CNN, CNN-Attention, BiLSTM, and BiLSTM-Attention using Macro Soft-F1 Loss and Macro Soft-F2 Loss can be seen in Section F, where the training loss, validation loss, training F1-score, and validation F1-score over the epochs of each model is shown. Looking at these graphs, it is clearly noticeable that no model suffered from underfitting or overfitting.

Table 6.19 shows the results for the Formspring dataset using Macro Soft-F1 Loss, while Table 6.20 shows the results for the Formspring dataset using RoBERTa. BiLSTM-Attention using GloVe Twitter seems to be the model with the best performance because it has better positive class F1-score, although it has the same macro average F1-score of CNN-Attention using GloVe Common Crawl, indicating that it predicts more positive class. Furthermore, the addition of the Attention layer seems to have a positive influence on the results. Surprisingly, RoBERTa has one of the worst results, but on the other hand has the best Recall with much higher values compared to the others. The reason behind this is the noisy nature of the data.

TABLE 6.19. Results for the Formspring dataset using Macro Soft-F1 Loss

GloVe	**Model**	**Macro Average**			**Positive Class**			
		Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
Twitter	CNN	**0.769**	0.716	0.738	**0.590**	0.464	0.519	**0.922**
	CNN-Attention	0.749	0.725	0.736	0.548	0.489	0.517	0.917
	BiLSTM	0.752	0.719	0.734	0.555	0.476	0.513	0.918
	BiLSTM-Attention	0.744	**0.737**	**0.740**	0.535	**0.519**	**0.527**	0.916
Common Crawl	CNN	0.740	0.719	0.729	0.531	0.481	0.505	0.914
	CNN-Attention	0.763	0.721	**0.740**	0.578	0.476	0.522	0.921
	BiLSTM	0.752	0.721	0.735	0.554	0.481	0.515	0.918
	BiLSTM-Attention	0.697	0.696	0.696	0.448	0.446	0.447	0.900

TABLE 6.20. Results for the Formspring dataset using RoBERTa

Threshold	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
Training set - 0.725	**0.702**	0.708	**0.705**	**0.456**	0.472	**0.464**	**0.901**
Default - 0.500	0.653	**0.771**	0.685	0.343	**0.669**	0.454	0.854
Testing set - 0.625	0.687	0.749	0.712	0.418	0.579	0.486	0.889

Table 6.21 shows the results for the OLID using the Macro Soft-F1 Loss, while Table 6.22 illustrate the results for the OLID using RoBERTa. Looking at the table, it can be seen that RoBERTa is the model with the best results, as one would expect. I can also highlight that BiLSTM using GloVe Common Crawl has the second best F1-score. Concerning the macro average, RoBERTa has the best scores with an obvious advantage over the other models. The other models have reasonable results compared to the

OffensEval 2019 results, but none is even close to having results as good as RoBERTa. Looking at the results of OffensEval 2019, where the OLID was used, the best result of this challenge was a macro average F1-score of 0.829 using BERT-base-uncased with default-parameters, a max sentence length of 64 and trained for 2 epochs. Second place, in turn, got a macro average F1-score of 0.815. RoBERTa used in this work achieved a macro average F1-score of 0.826, indicating a negative difference of 0.003 for the first place and a positive difference of 0.011 for the second place.

TABLE 6.21. Results for the OLID using Macro Soft-F1 Loss

GloVe	**Model**	**Macro Average**			**Positive Class**			
		Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
Twitter	CNN	0.759	0.762	0.760	0.649	0.662	0.656	0.806
	CNN-Attention	0.764	0.766	0.765	0.658	0.667	0.663	0.810
	BiLSTM	0.713	0.750	0.721	0.543	**0.742**	0.627	0.753
	BiLSTM-Attention	0.761	0.769	0.765	0.649	0.679	0.664	0.808
Common Crawl	CNN	0.776	**0.795**	0.784	0.658	0.738	0.695	0.820
	CNN-Attention	0.780	0.792	0.785	0.671	0.721	0.695	0.823
	BiLSTM	**0.801**	0.791	**0.796**	**0.722**	0.683	**0.702**	**0.838**
	BiLSTM-Attention	0.762	0.787	0.772	0.631	**0.742**	0.682	0.807

TABLE 6.22. Results for the OLID using RoBERTa

Threshold	**Macro Average**			**Positive Class**			
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Acc**
Training set - 0.475	0.826	**0.817**	0.821	0.757	0.725	0.740	0.858
Testing set/Default - 0.500	**0.837**	0.816	**0.826**	**0.783**	0.708	**0.744**	**0.864**

The results for the SOLID using the Macro Soft-F1 Loss are shown in Table 6.23. Table 6.24 reports the results for the SOLID using RoBERTa. Once again, RoBERTa is the model with the best performance, outperforming all other models in terms of macro average. However, it should be noted that CNN using GloVe Twitter and CNN-Attention using Common Crawl show remarkable results, being even the second best models, presenting similar scores. All these models have notable results compared to the OffensEval 2020 results. The results of OffensEval 2020, where the SOLID was used, show that the best result was a macro average F1-score of 0.9204 and compared to the macro average F1-score of RoBERTa used in this work, it can be seen that RoBERTa outperformed that result.

TABLE 6.23. Results for the SOLID using Macro Soft-F1 Loss

GloVe	Model	Macro Average			Positive Class			
		Prec	Rec	F1	Prec	Rec	F1	Acc
Twitter	CNN	**0.890**	**0.936**	**0.908**	**0.792**	0.971	**0.872**	**0.921**
	CNN-Attention	0.881	0.933	0.900	0.772	0.977	0.863	0.914
	BiLSTM	0.882	0.921	0.897	0.789	0.938	0.857	0.913
	BiLSTM-Attention	0.852	0.917	0.869	0.707	**0.994**	0.826	0.884
Common Crawl	CNN	0.880	0.935	0.899	0.764	0.988	0.862	0.912
	CNN-Attention	0.889	**0.936**	0.907	0.791	0.971	**0.872**	**0.921**
	BiLSTM	0.872	0.926	0.891	0.757	0.971	0.851	0.905
	BiLSTM-Attention	0.883	0.935	0.902	0.776	0.980	0.866	0.916

TABLE 6.24. Results for the SOLID using RoBERTa

Threshold	Macro Average			Positive Class			
	Prec	Rec	F1	Prec	Rec	F1	Acc
Training set - 0.475	0.898	0.946	0.916	0.803	**0.986**	0.885	0.929
Default - 0.500	**0.902**	**0.948**	**0.920**	**0.812**	0.983	**0.889**	**0.932**
Testing set - 0.625	0.914	0.944	0.927	0.846	0.956	0.898	0.939

As already explained, in order to solve the imbalanced classification problem, Macro Soft-F2 Loss was used. Table 6.25 focus on the results for the Formspring dataset, Table 6.26 for the OLID, and Table 6.27 for the SOLID, all using the Macro Soft-F2 Loss. In general, compared to Macro Soft-F1 Loss, the results show that Recall increased, and on the other hand Precision, F1-score, and Accuracy decreased. As discussed earlier, the use of F2-score causes an increase in Recall and a decrease in Precision. BiLSTM-Attention using GloVe Twitter has the best Precision and F1-score, the third best Recall and the best macro average Accuracy for the Formspring dataset. For the OLID, CNN using GloVe Common Crawl has all the best evaluation metrics in terms of macro average and the best positive class F1-score. For the SOLID, BiLSTM-Attention has the best positive class Recall and positive class F1-score, as well as the best macro average Recall and the same macro average F1-score of CNN and BiLSTM, both using GloVe Common Crawl.

TABLE 6.25. Results for the Formspring dataset using Macro Soft-F2 Loss

GloVe	Model	Macro Average			Positive Class			
		Prec	Rec	F1	Prec	Rec	F1	Acc
Twitter	CNN	0.711	0.753	0.729	0.465	0.571	0.513	0.902
	CNN-Attention	0.712	0.729	0.720	0.472	0.515	0.493	0.904
	BiLSTM	0.704	0.745	0.722	0.453	0.558	0.500	0.899
	BiLSTM-Attention	**0.718**	0.747	**0.731**	**0.481**	0.554	**0.515**	**0.905**
Common Crawl	CNN	0.713	0.720	0.716	0.477	0.494	0.485	**0.905**
	CNN-Attention	0.706	0.731	0.718	0.460	0.524	0.490	0.901
	BiLSTM	0.698	0.740	0.716	0.443	0.549	0.490	0.897
	BiLSTM-Attention	0.685	**0.759**	0.713	0.411	**0.605**	0.490	0.886

TABLE 6.26. Results for the OLID using Macro Soft-F2 Loss

GloVe	Model	Macro Average			Positive Class			
		Prec	Rec	F1	Prec	Rec	F1	Acc
Twitter	CNN	0.724	0.766	0.732	0.550	0.779	0.645	0.760
	CNN-Attention	0.738	0.766	0.747	**0.592**	0.725	0.652	0.784
	BiLSTM	0.704	0.747	0.709	0.518	0.771	0.620	0.736
	BiLSTM-Attention	0.719	0.763	0.726	0.540	0.783	0.639	0.753
Common Crawl	CNN	**0.746**	**0.788**	**0.757**	0.584	0.796	**0.674**	**0.785**
	CNN-Attention	0.737	0.774	0.747	0.577	0.767	0.658	0.778
	BiLSTM	0.707	0.756	0.698	0.494	**0.850**	0.625	0.715
	BiLSTM-Attention	0.733	0.783	0.737	0.545	0.838	0.660	0.759

TABLE 6.27. Results for the SOLID using Macro Soft-F2 Loss

GloVe	Model	Macro Average			Positive Class			
		Prec	Rec	F1	Prec	Rec	F1	Acc
Twitter	CNN	0.865	0.927	0.884	0.732	0.994	0.843	0.897
	CNN-Attention	0.869	0.927	0.888	0.747	0.981	0.848	0.902
	BiLSTM	0.869	0.930	0.888	0.742	0.993	0.849	0.902
	BiLSTM-Attention	0.863	0.926	0.882	0.728	0.995	0.841	0.896
Common Crawl	CNN	**0.876**	0.931	**0.894**	0.760	0.981	0.856	0.908
	CNN-Attention	0.869	0.929	0.887	0.740	0.993	0.848	0.901
	BiLSTM	**0.876**	0.929	**0.894**	**0.763**	0.973	0.856	**0.909**
	BiLSTM-Attention	0.875	**0.935**	**0.894**	0.751	**0.996**	**0.857**	0.907

6.3. Summary of Experiments

This section summarises the results of the experiments carried out in this work. Table 6.28 shows the most relevant experiments for each dataset with the respective macro average F1-score. The experiments E2, E10 and E15 are the experiments with the best macro average F1-score for the Formspring dataset, OLID, and SOLID, respectively. Across all datasets, the best performing experiments were those using Deep Learning approaches,

being in line with the works reported in Chapter 3. Therefore, according to this fact, the Deep Learning approach is the most suitable to solve this problem. The Formspring dataset is the most imbalanced dataset, which explains the lowest relative results attained, when compared with the other two datasets, meaning that the greater the imbalance, the worse the result. In general, RoBERTa is the most suitable model to solve these problems, presenting state-of-the-art level results.

TABLE 6.28. Selected experiments for each dataset and macro average F1-score associated

Dataset	**Experiments**	**F1**
Formspring	E1. Logistic Regression, F1-score optimal threshold, TF-IDF	0.732
	E2. BiLSTM-Attention, GloVe Twitter, Macro Soft-F1 Loss	**0.740**
	E3. SVM, F2-score optimal threshold, single weight per document	0.714
	E4. BiLSTM-Attention, GloVe Twitter, Macro Soft-F2 Loss	0.731
	E5. RoBERTa, Training set threshold	0.705
OLID	E6. Logistic Regression, F1-score optimal threshold, TF-IDF	0.745
	E7. BiLSTM, GloVe Common Crawl, Macro Soft-F1 Loss	0.796
	E8. SVM, SMOTEENN, single weight per document	0.714
	E9. CNN, GloVe Common Crawl, Macro Soft-F2 Loss	0.757
	E10. RoBERTa, Default threshold	**0.826**
SOLID	E11. Logistic Regression, F1-score optimal threshold, TF-IDF	0.899
	E12. CNN, GloVe Twitter, Macro Soft-F1 Loss	0.908
	E13. Logistic Regression, SMOTEENN, single weight per document	0.856
	E14. BiLSTM-Attention, GloVe Common Crawl, Macro Soft-F2 Loss	0.894
	E15. RoBERTa, Default threshold	**0.920**

CHAPTER 7

Conclusions, Limitations, and Future Work

This final chapter presents the conclusions, the limitations identified during the course of this work, and a proposal for future work.

7.1. Conclusions

Cyberbullying, a phenomenon deeply related to offensive language and hate speech, has increasingly emerged as a social problem, prompting the need to develop systems capable of detecting offensive texts in social media. The detection of these phenomena is a complex and very hard task because in addition to being an extremely imbalanced problem, due to the relatively low number of samples of the positive class, it also has a high degree of subjectivity. In this work, three datasets were used, namely the Formspring dataset, OLID, and SOLID. The data source of the Formspring dataset is the Formspring social media, while in the OLID and SOLID is Twitter. The OLID was used in "SemEval 2019 challenge: OffensEval 2019 – Identifying and Categorizing Offensive Language in Social Media (Task 6)" and SOLID was used in "SemEval 2020 challenge: OffensEval 2020 — Multilingual Offensive Language Identification in Social Media (Task 12)". Two main approaches were proposed and developed in this work: classical Machine Learning approach and Deep Learning approach.

In the classical Machine Learning approach, I proposed a specific pre-processing and Feature Engineering stage with several steps: grammatical features, bad words features, profanity features, and sentiment features, culminating in the application of two different document representation approaches to generate the inputs used by three algorithms: SVM, Logistic Regression, and Random Forest. Threshold-Moving and SMOTEENN mitigated the imbalanced classification problem, which is usually a great obstacle in such classification problems. The use of Threshold-Moving to find F2-score optimal threshold or SMOTEENN depends on whether it is preferable or not to sacrifice Precision and Accuracy in order to increase Recall, correctly predicting more instances of the positive class. The results show that these approaches are suitable to detect offensive texts on social media and can be a starting point for building a more robust system.

Deep Learning has been increasingly used to classify offensive texts on social media. In the Deep Learning approach, I proposed five different DNN architectures for tackling this problem: CNN, CNN-Attention, BiLSTM, BiLSTM-Attention, and RoBERTa. These models used two pre-trained word vectors: GloVe Twitter and GloVe Common Crawl, as well as an experimental setup that involved the treatment of OOV words, CLR to provide better convergence, Macro Soft-F1 Loss function to optimize performance, and Macro

Soft-F2 Loss function to deal with imbalanced classification problem. RoBERTa-base model, pre-trained on 58 million tweets and finetuned for offensive language identification, was in general the model with the best results, achieving a remarkable performance. For OffensEval 2019, RoBERTa obtained the second best macro average F1-score, while in OffensEval 2020 it achieved the best macro average F1-score. In summary, the experimental results showed that the proposed DNN are suitable for detecting offensive texts, achieving state-of-the-art results.

As previously mentioned, the Deep Learning approach was the one that obtained the best results, so I conclude that, according to the experiences done throughout this work, the Deep Learning approach is the most suitable to solve this problem. It also important to refer that the pre-trained word embeddings models and DNN can be applied in one domain and be transferred to another, while maintaining a similar performance.

7.2. Limitations and Future Work

The complexity, size and imbalance of the datasets were the main challenges in this work. Another aspect to take into account in the classical Machine Learning approach is that the used toolkits are not error free and can have a significant impact on the generation of features for the the single weight document representation. Concerning the Deep Learning approach, the OOV words, the dependence of WordSegment for word segmentation and Spello for spelling correction are the most important limitations to mention.

As future work it might be interesting to use other different DNN architectures, as well as other pre-trained word embeddings models, in particular fastText. Finally, it could be advantageous to apply these approaches to other larger datasets in order to understand the real capacity and generalisation of these approaches.

References

[1] G. Gredler, "Olweus, d. (1993).bullying at school: What we know and what we can do. malden, ma: Blackwell publishing, 140 pp." *Psychology in The Schools - PSYCHOL SCH*, vol. 40, pp. 699–700, 2003.

[2] P. K. Smith, J. Mahdavi, M. Carvalho, S. Fisher, S. Russell, and N. Tippett, "Cyberbullying: its nature and impact in secondary school pupils," *Journal of Child Psychology and Psychiatry*, vol. 49, no. 4, pp. 376–385, 2008.

[3] What is cyberbullying | StopBullying.gov. [Online]. Available: https://www.stopbullying.gov/cyberbullying/what-is-it

[4] R. Zhao, A. Zhou, and K. Mao, "Automatic detection of cyberbullying on social networks based on bullying features," in *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ser. ICDCN '16. New York, NY, USA: Association for Computing Machinery, 2016.

[5] K. Nalini and L. Jaba Sheela, "Classification using Latent Dirichlet Allocation with Naive Bayes Classifier to detect Cyber Bullying in Twitter," *Indian Journal of Science and Technology*, vol. 9, no. 28, 2016.

[6] Maioria dos estudantes foi vítima de cyberbullying durante a pandemia. [Online]. Available: https://www.iscte-iul.pt/noticias/1706/maioria-estudantes-vitima-de-cyberbullying-durante-pandemia

[7] S. Hinduja and J. W. Patchin, "Bullying, cyberbullying, and suicide," *Archives of suicide research: International Academy for Suicide Research*, vol. 14, no. 3, pp. 206–221, 2010.

[8] Bullying and suicide. [Online]. Available: http://www.bullyingstatistics.org/content/bullying-and-suicide.html

[9] M. Al-Hashedi, L.-K. Soon, and H.-N. Goh, "Cyberbullying Detection Using Deep Learning and Word Embeddings: An Empirical Study," in *Proceedings of the 2019 2nd International Conference on Computational Intelligence and Intelligent Systems*. Bangkok Thailand: ACM, 2019, pp. 17–21.

[10] V. Nahar, X. Li, C. Pang, and Y. Zhang, "Cyberbullying Detection based on Text-Stream Classification," in *AusDM 2013*, vol. 146, 2013, p. 9.

[11] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, 2000, pp. 29–39.

[12] R. Sharda, D. Delen, and E. Turban, *Business Intelligence: A Managerial Perspective on Analytics*. Pearson, 2013.

[13] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. R. Shearer, and R. Wirth, "Crisp-dm 1.0: Step-by-step data mining guide," 2000.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[15] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997.

[16] J. Toutouh, J. Arellano-Verdejo, and E. Alba, "Bipred: A bilevel evolutionary algorithm for prediction in smart mobility," *Sensors*, vol. 18, p. 4123, 2018.

[17] F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781 [cs.CL]*, 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv:1310.4546 [cs, stat]*, 2013.

[20] GloVe: Global vectors for word representation. [Online]. Available: https://nlp.stanford.edu/projects/glove/

[21] M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong, "Improving Cyberbullying Detection with User Context," in *Advances in Information Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7814, pp. 693–696.

[22] M. Dadvar, F. de Jong, R. Ordelman, and D. Trieschnigg, "Improved cyberbullying detection using gender information," in *Proceedings of the Twelfth Dutch-Belgian Information Retrieval Workshop (DIR 2012)*. Belgium: Universiteit Gent, 2012, pp. 23–25.

[23] A. Saravanaraj, J. I. Sheeba, and S. P. Devaneyan, "Automatic Detection of Cyberbullying From Twitter," *International Journal of Computer Science and Information Technology*, p. 6, 2016.

[24] M. Di Capua, E. Di Nardo, and A. Petrosino, "Unsupervised cyber bullying detection in social networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. Cancun: IEEE, 2016, pp. 432–437.

[25] M. Dittenbach, D. Merkl, and A. Rauber, "The growing hierarchical self-organizing map," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 6, 2000, pp. 15–19 vol.6.

[26] C. Van Hee, G. Jacobs, C. Emmery, B. Desmet, E. Lefever, B. Verhoeven, G. De Pauw, W. Daelemans, and V. Hoste, "Automatic detection of cyberbullying in social media text," *PLOS ONE*, vol. 13, no. 10, p. e0203794, 2018.

[27] H. Rosa, N. Pereira, R. Ribeiro, P. Ferreira, J. Carvalho, S. Oliveira, L. Coheur, P. Paulino, A. Veiga Simão, and I. Trancoso, "Automatic cyberbullying detection: A systematic review," *Computers in Human Behavior*, vol. 93, pp. 333–345, 2019.

[28] J. Hani, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed, "Social Media Cyberbullying Detection using Machine Learning," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, 2019.

[29] H. Sanchez and S. Kumar, "Twitter Bullying Detection," in *NSDI'12*. Berkeley, CA: USENIX Association, 2012, p. 15–15.

[30] J. L. Bigelow, A. Edwards (Kontostathis), and L. Edwards, "Detecting Cyberbullying using Latent Semantic Indexing," in *Proceedings of the First International Workshop on Computational Methods for CyberSafety - CyberSafety'16*. Indianapolis, IN, USA: ACM Press, 2016, pp. 11–14.

[31] B. S. Nandhini and J. Sheeba, "Online Social Network Bullying Detection Using Intelligence Techniques," *Procedia Computer Science*, vol. 45, pp. 485–492, 2015.

[32] H. Rosa, J. P. Carvalho, P. Calado, B. Martins, R. Ribeiro, and L. Coheur, "Using Fuzzy Fingerprints for Cyberbullying Detection in Social Networks," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Rio de Janeiro: IEEE, 2018, pp. 1–7.

[33] E. Raisi and B. Huang, "Cyberbullying Identification Using Participant-Vocabulary Consistency," *arXiv:1606.08084 [cs.CL]*, 2016.

[34] Sherly and B. R. Jeetha, "Supervised feature selection based extreme learning machine (SFS-ELM) classifier for cyber bullying detection in twitter," *International Journal of Scientific and Research Publications*, vol. 7, no. 7, p. 7, 2017.

[35] Noviantho, S. M. Isa, and L. Ashianti, "Cyberbullying Classification using Text Mining," in *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*, 2017, pp. 241–246.

[36] X. Zhang, J. Tong, N. Vishwamitra, E. Whittaker, J. P. Mazer, R. Kowalski, H. Hu, F. Luo, J. Macbeth, and E. Dillon, "Cyberbullying Detection with a Pronunciation Based Convolutional Neural Network," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Anaheim, CA, USA: IEEE, 2016, pp. 740–745.

[37] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very Deep Convolutional Networks for Text Classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 1107–1116.

[38] R. Zhao and K. Mao, "Cyberbullying Detection Based on Semantic-Enhanced Marginalized Denoising Auto-Encoder," *IEEE Transactions on Affective Computing*, vol. 8, no. 3, pp. 328–339, 2017.

[39] Q. Huang, D. Inkpen, J. Zhang, and D. Van Bruwaene, "Cyberbullying Intervention Interface Based on Convolutional Neural Networks," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 42–51.

[40] M. A. Al-Ajlan and M. Ykhlef, "Deep Learning Algorithm for Cyberbullying Detection," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, 2018.

[41] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1322–1328.

[42] H. Rosa, D. Matos, R. Ribeiro, L. Coheur, and J. P. Carvalho, "A "Deeper" Look at Detecting Cyberbullying in Social Networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, 2018, pp. 1–8.

[43] M. Dadvar and K. Eckert, "Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study," *arXiv:1812.08046 [cs, stat]*, p. 13, 2018.

[44] S. Agrawal and A. Awekar, "Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms," in *Advances in Information Retrieval*, G. Pasi, B. Piwowarski, L. Azzopardi, and A. Hanbury, Eds. Cham: Springer International Publishing, 2018, vol. 10772, pp. 141–153.

[45] P. Kapil', A. Ekbal', and D. Das, "Investigating Deep Learning Approaches for Hate Speech Detection in Social Media," *arXiv:2005.14690 [cs.CL]*, p. 12, 2020.

[46] Z. Zhang, D. Robinson, and J. Tepper, "Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network," in *Proceedings of WWW'2018*, Lyon, France, 2018.

[47] T. Van Huynh, V. D. Nguyen, K. Van Nguyen, N. L.-T. Nguyen, and A. G.-T. Nguyen, "Hate Speech Detection on Vietnamese Social Media Text using the Bi-GRU-LSTM-CNN Model," *arXiv:1911.03644 [cs.CL]*, 2019.

[48] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media," *arXiv:1910.12574 [cs.CL]*, 2019.

[49] R. Ong, "Offensive Language Analysis using Deep Learning Architecture," *arXiv:1903.05280 [cs.CL]*, 2019.

[50] OffensEval shared task. [Online]. Available: https://sites.google.com/site/offensevalsharedtask/home

[51] International workshop on semantic evaluation 2019 - SemEval-2019. [Online]. Available: https://alt.qcri.org/semeval2019/

[52] International workshop on semantic evaluation 2020 - SemEval-2020. [Online]. Available: https://alt.qcri.org/semeval2020/

[53] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval)," in *Proceedings of the 13th International Workshop on Semantic*

Evaluation. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019, pp. 75–86.

[54] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," in *Proceedings of the 2019 Conference of the North*. Association for Computational Linguistics, 2019, pp. 1415–1420.

[55] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and c. ÇöÇöltekin, "SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020)," *arXiv:2006.07235 [cs.CL]*, 2020.

[56] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri, and P. Nakov, "A large-scale semi-supervised dataset for offensive language identification," *arXiv:2004.14454 [cs]*, 2020.

[57] P. Liu, W. Li, and L. Zou, "NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 2019, pp. 87–91.

[58] W. Dai, T. Yu, Z. Liu, and P. Fung, "Kungfupanda at SemEval-2020 task 12: BERT-based multi-task learning for offensive language detection," *arXiv:2004.13432 [cs.CL]*, 2020.

[59] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805 [cs.CL]*, 2019.

[60] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv:1907.11692 [cs.CL]*, 2019.

[61] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 8440–8451.

[62] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *arXiv:1909.11942 [cs.CL]*, 2020.

[63] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *Technical report, OpenAI*, p. 24, 2019.

[64] K. Reynolds, A. Kontostathis, and L. Edwards, "Using Machine Learning to Detect Cyberbullying," in *2011 10th International Conference on Machine Learning and Applications and Workshops*. Honolulu, HI: IEEE, 2011, pp. 241–244.

[65] Complete list of HTML entities - FreeFormatter.com. [Online]. Available: https://www.freeformatter.com/html-entities.html

[66] Bad words list. [Online]. Available: https://www.cs.cmu.edu/~biglou/resources/bad-words.txt

[67] Full list of bad words and swear words banned by google. [Online]. Available: https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/

[68] Banned word list. [Online]. Available: http://www.bannedwordlist.com/lists/swearWords.txt

[69] List of dirty naughty obscene and otherwise bad words. [Online]. Available: https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

[70] RobertJGabriel/google-profanity-words. [Online]. Available: https://github.com/RobertJGabriel/Google-profanity-words/blob/master/list.txt

[71] Internet slang dataset. [Online]. Available: https://floatcode.wordpress.com/2015/11/28/internet-slang-dataset/

[72] R. S. R. Srivastava Aman, "spello," https://github.com/hellohaptik/spello, 2020.

[73] V. Zhou. profanity-check: A fast, robust library to check for offensive language in strings. [Online]. Available: https://github.com/vzhou842/profanity-check

[74] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.

[75] A. Power, A. Keane, B. Nolan, and B. O'Neill, "A lexical database for public textual cyberbullying detection," *Revista de Lenguas para Fines Específicos*, 2017.

[76] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st ed. O'Reilly Media, Inc., 2009.

[77] Difflib — helpers for computing deltas — python 3.9.1 documentation. [Online]. Available: https://docs.python.org/3/library/difflib.html

[78] Fuzzywuzzy. [Online]. Available: https://github.com/seatgeek/fuzzywuzzy

[79] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals." *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[80] G. Jenks. grantjenks/python-wordsegment. [Online]. Available: https://github.com/grantjenks/python-wordsegment

[81] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[82] Zhi-Hua Zhou and Xu-Ying Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[83] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, p. 20–29, 2004.

[84] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, 2002.

[85] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[86] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv:1408.5882 [cs.CL]*, 2014.

[87] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016, pp. 1480–1489.

[88] Q. Zhou and H. Wu, "NLP at IEST 2018: BiLSTM-attention and LSTM-attention via soft voting in emotion classification," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 189–194.

[89] L. N. Smith, "Cyclical learning rates for training neural networks," *arXiv:1506.01186 [cs.CL]*, 2017.

[90] A. Maiza. The unknown benefits of using a soft-f1 loss in classification systems. [Online]. Available: https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d

[91] Tensorflow. [Online]. Available: https://github.com/tensorflow/tensorflow

[92] Keras. [Online]. Available: https://github.com/keras-team/keras-io

[93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv:1706.03762 [cs.CL]*, 2017.

[94] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152.

[95] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016, pp. 1715–1725.

[96] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics,

2018, pp. 66–71.

[97] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, and L. Neves, "TweetEval: Unified benchmark and comparative evaluation for tweet classification," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020, pp. 1644–1650.

[98] cardiffnlp/twitter-roberta-base-offensive · hugging face. [Online]. Available: https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive

[99] P. Bhuvaneswari and J. SatheeshKumar, "Support vector machine technique for eeg signals," *International Journal of Computer Applications*, vol. 63, pp. 1–5, 2013.

[100] C. Fan, Z. Xie, Y. Liu, C. Li, and H. Liu, "Adaptive controller based on spatial disturbance observer in a microgravity environment," *Sensors*, vol. 19, p. 4759, 2019.

[101] A. Gelzinis, A. Verikas, E. Vaiciukynas, M. Bacauskiene, J. Minelga, M. Hållander, V. Uloza, and E. Padervinskis, "Exploring sustained phonation recorded with acoustic and contact microphones to screen for laryngeal disorders," in *2014 IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE)*, 2014, pp. 125–132.

[102] J. J. Gago, V. Vasco, B. Łukawski, U. Pattacini, V. Tikhanoff, J. Victores, and C. Balaguer, "Sequence-to-sequence natural language to humanoid robot sign language," *arXiv:1907.04198 [cs.RO]*, p. 13, 2019.

[103] A. E.-D. Mousa and B. Schuller, "Deep bidirectional long short-term memory recurrent neural networks for grapheme-to-phoneme conversion utilizing complex many-to-many alignments," in *Interspeech 2016*, 2016.

Appendices

A. Evaluation Metrics

Before having the knowledge about evaluation metrics, it is necessary to understand the definitions of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN):

- TP: The number of instances where the model correctly predicts the positive class;
- FP: The number of instances where the model incorrectly predicts the positive class;
- TN: The number of instances where the model correctly predicts the negative class;
- FN: The number of instances where the model incorrectly predicts the negative class.

Having understood these definitions, already it is possible explorer the Confusion Matrix. The Confusion Matrix is a table that allows visualize the performance of the models, the number of right and wrong predictions for each class, through TP, FP, TN and FN discussed above. Figure A1 shows an illustration of Confusion Matrix.

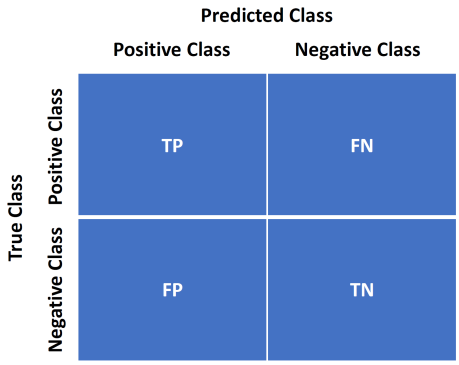


FIGURE A1. Confusion Matrix

There are evaluation metrics for Classification, Regression, and Clustering, but as the problem of this work is a binary classification problem, only the main classification evaluation metrics were used. These metrics are Precision, Recall, F1-score, and Accuracy. Accuracy is the best known and most used evaluation metric, being defined as the fraction of correct predictions by the model. For binary classification, Accuracy is calculated through the Equation A.1.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{A.1}$$

Accuracy is a more appropriate evaluation metric to be used when the datasets are balanced, but not in imbalanced datasets, because can be happen that the model predict correctly the most or even all cases of negative class (majority class) and predict wrong the most or even all cases of positive class (minority class). In this case, Accuracy will

be quite high and the percentage of imbalance between the classes is what defines how high Accuracy will be. For this reason, F1-score are the most appropriated metric to be taken into consideration for imbalanced classification problem. Precision is defined as the fraction of correctly identified as positive class out of all predicted as positive class and it is calculated through the Equation A.2.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{A.2}$$

Precision has the disadvantage of not taking into account FN and the advantage of minimizing FP. On the other hand, Recall is defined as the fraction of correctly identified as positive class out of all instances of positive class. Recall is calculated through the Equation A.3.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{A.3}$$

Recall has the disadvantage of not taking into account FP and the advantage is to minimize the FN. F1-score is a harmonic mean of Precision and Recall, allowing to combine these two metrics and capture both properties. The ability to have Precision and Recall equally combined and balanced in terms of its importance, makes F1-score a special metric. F1-score is a particular case of Fβ-score and the most common metric used in these particular cases. The Fβ-score is an abstraction of F1-score, F2-score, among others, where the balance of Precision and Recall is controlled by a coefficient called β. F1-score, e.g., has the β value of 1, while F2-score has the β value of 2. The Fβ-score is calculated through the Equation A.4:

$$\text{F}\beta\text{-score} = \frac{(1 + \beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{A.4}$$

While F1-score focuses on balancing the importance of both metrics, F2-score focuses on decreasing the importance of Precision and increasing the importance of Recall and, consequently, pays more attention to minimizing FN than to minimizing FP, predicting more correctly the texts related to the positive class, rather than texts related to the negative class.

B. Classical Machine Learning Algorithms

This section is intended to introduce basic concepts about classical Machine Learning algorithms, such as SVM, Logistic Regression and Random Forest.

B.1. SVM

SVM is an algorithm that can solve linear and non linear problems. It basically takes the data and creates a line or hyperplane that separates this data into classes, in this case into two classes, because this problem is a binary classification problem. It aims to make a a decision boundary in such a manner that the separation between the two classes is as broad as possible. To find the best hyperplane, it is necessary to find the support vectors, which are the points closest to the line of both classes, compute the distance between the line and support vectors, being this distance called margin. The goal is to maximize this margin and the hyperplane with the maximum margin is considered the optimal hyperplane. Figure B2 illustrates the case where data is linearly separable.

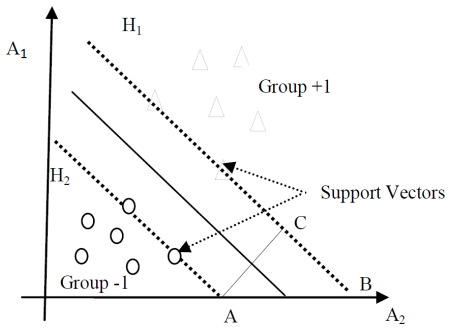


FIGURE B2. SVM example when data is linearly separable [99].

In case the data is non linearly separable, it is necessary to convert this data into linearly separable data in higher dimensions. To do this, a new dimension is added, which is called the z-axis and is the square of the distance of the point from the origin:

$$z = x^2 + y^2 \tag{B.1}$$

Figure B3 illustrates the case where data is non linearly separable.

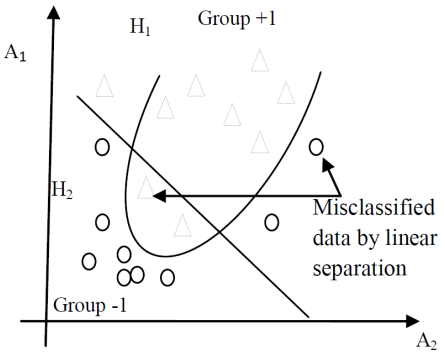


FIGURE B3. SVM example when data is non linearly separable [99].

There are several parameters that need to be defined when creating this model and the most important are C, Kernel and Gamma. C tries to control the trade-off between smooth decision boundary and the correct classification of the training points. Kernel is a set of mathematical functions that take the input and transform it into the required form, depending on the type of functions. The main functions are Linear, Polynomial, RBF and Sigmoid. Linear kernel is used when the data is linearly separable and is mostly used in text classification. For a point x and another point y, the equation of Linear kernel is defined in Equation B.2:

$$k(x, y) = x^T y \tag{B.2}$$

The equation of Polynomial kernel is defined in Equation B.3, where d which is the degree of the polynomial, slope α and the constant term c can be adjusted:

$$k(x, y) = (\alpha x^T y + c)^d \tag{B.3}$$

Sigmoid kernel also known as the Hyperbolic Tangent kernel and as Multilayer Perceptron kernel originated in the DNN field, where the Sigmoid function can be used as an activation function and when SVM use this kernel is equivalent to a two-layer DNN. The Equation B.4 specify the Sigmoid kernel, where α has a common value of $\frac{1}{N}$, being N the data dimension:

$$k(x, y) = \tanh(\alpha x^T y + c) \tag{B.4}$$

RBF kernel is the most commonly used kernel in SVM and it is used when there is no prior knowledge about the data. RBF kernel is characterized in Equation B.5, where $\| x{-}y \|^2$ is the squared Euclidean distance between x and y:

$$k(x, y) = e(-\frac{\| x - y \|^2}{2\sigma^2}) \tag{B.5}$$

Gamma is only used when using RBF kernel, so Gamma is a parameter of the RBF kernel. Gamma can be defined as the curvature of a decision boundary and can be used as σ in Equation B.6:

$$\text{gamma} = \frac{1}{2\sigma^2} \tag{B.6}$$

B.2. Logistic Regression

Logistic Regression as opposed to Linear Regression uses a more complex cost function instead of a simple linear function. The hypothesis of Logistic Regression tends to constrain the cost function between 0 and 1 and is represented by Equation B.7, where x is the observation of index i and β is the regression coefficient vector:

$$h(x_i) = f(\beta^T x_i) = \frac{1}{1 + e^{-\beta^T x_i}} \tag{B.7}$$

This function is called Sigmoid function or also known as logistic function and its purpose is to transform predicted values into probabilities, transforming any real value z into a value y between 0 and 1:

$$f(z) = \frac{1}{1 + e^{-(z)}} \tag{B.8}$$

It is a function that when plotted on a graph resembles an S-shaped curve and can be seen in Figure B4:

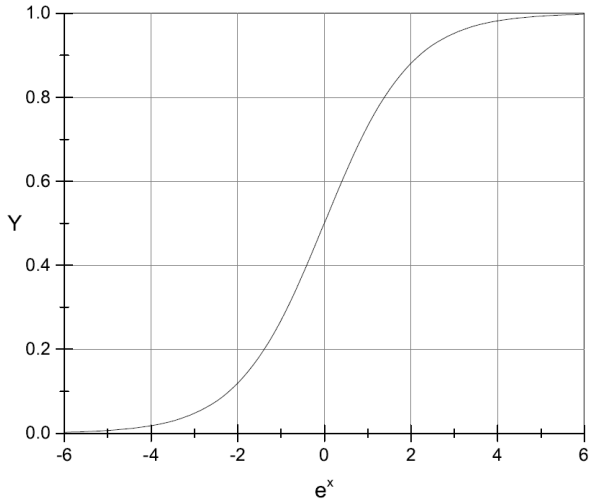


FIGURE B4. Logistic function [100].

The cost function of Logistic Regression is proportional to inverse of probability of parameters and produce the cost value or error $J(\beta)$:

$$J(\beta) = \sum_{i=1}^{n} -y \log(h(x_i)) - (1 - y_i) \log(1 - h(x_i)) \tag{B.9}$$

In order to minimize the cost value, Gradient Descent is used. Normally, an analogy is used to describe Gradient Descent, in which one imagines the top of a mountain valley and the goal is to reach the bottom of the hill. It is necessary to run Gradient Descent function on each parameter, where y is the response vector, $h(x)$ is the predicted response vector, x is the vector representing the observation values for j feature and α is the learning rate that has to be defined:

ALGORITHM .1. Gradient Descent algorithm

Repeat {

$\beta_j := \beta_j - \alpha \sum_{i=1}^{n} (h(x_i) - y_i) x_{ij}$

(Simultaneously update all β_j)

}

Gradient Descent is one of the many algorithms that has the goal of minimizing the cost value. Other algorithms are more advanced and some of them are the following: Newton-CG, L-BFGS, LIBLINEAR, SAG, and SAGA.

B.3. Random Forest

To understand Random Forest, it is necessary to understand the Decision Trees because Random Forest randomly selects observations and features to build multiple Decision Trees, trains them with the Bagging method and applies majority voting the results to get a more stable prediction.

The Decision Trees can be defined as a set of decision rules, learning from the data to approximate a sine curve using these rules. The theory of Decision Trees is to split the dataset into smaller subsets based on the features until a subset is reached that incorporates specifications that fit a respective label. The root nodes are represented by the features and the leaf nodes by the results, where each feature becomes a root node and each result becomes a leaf node. For each split it is necessary to measure its quality and to do this Gini Impurity or Information Gain are used. Gini Impurity or also know as Gini Index measures the probability that a specific observation is misclassified when randomly selected and is calculated as follows, where C is the number of classes and p is the probability:

$$\text{Gini} = 1 - \sum_{i}^{C} (p_i)^2 \tag{B.10}$$

Entropy is a measure used to check the homogeneity and impurity of the data, referring to disorder or uncertainty and is an important measure for the calculation of Information Gain. Entropy is calculated using Equation B.11:

$$\text{Entropy} = -\sum_{i}^{C} p_i \log_2 p_i \tag{B.11}$$

In both criteria, the optimal split is chosen by the features with less value. Information Gain measures which feature provides the maximum information with the goal of reducing the amount of Entropy from the root node to the leaf nodes. Information Gain is calculated using the Equation B.12:

$$\text{IG} = \text{Entropy(parent)} - [\text{weighted average}]\,\text{Entropy(children)} \tag{B.12}$$

There is a method used in Decision Trees to avoid the overfitting called Pruning, reducing the size of the learning tree by removing nodes without reducing performance, limiting tree depth. Usually, Pre-Pruning is used, a type of Pruning that involves defining the parameters before the Decision Tree is built. These parameters are the maximum tree

depth, the maximum number of terminal nodes, the minimum samples for a node split and the maximum number of features. Figure B5 illustrate the Random Forest architecture:

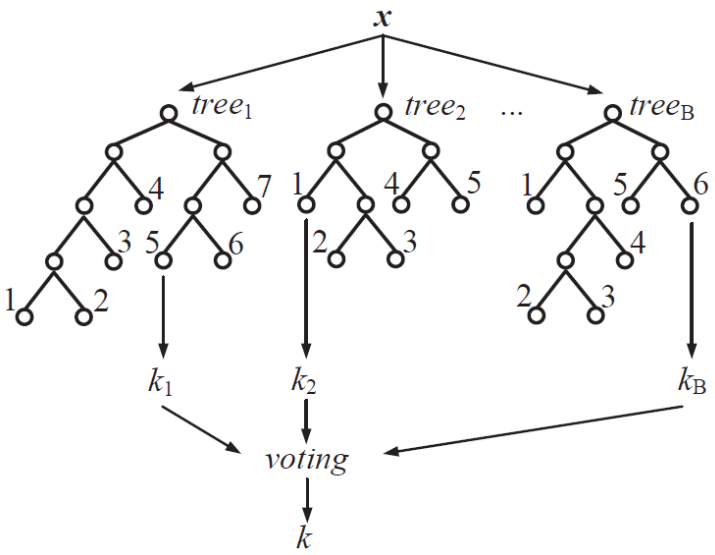


FIGURE B5. Random Forest architecture [101].

C. Neural Networks

In this section, the basic concepts about CNN, LSTM, and BiLSTM are presented.

C.1. CNN

Initially, CNN was created for Computer Vision, showing a good performance. In addition to being successful in these tasks, it was also successful in NLP tasks, having a strong feature representation capability and training using a Graphics Processing Unit (GPU) allows for greater time efficiency compared to other Deep Learning models. The main operations in CNN are convolutions, pooling and activation. The main block is a convolutional layer that performs convolutions in the inputs with different filters, producing an activation map of the responses of the filters used. Pooling layer are frequently used between convolutional layers to deal with overfitting. The CNN architecture, shown in Figure C6, is a model architecture with two channels for an example sentence:

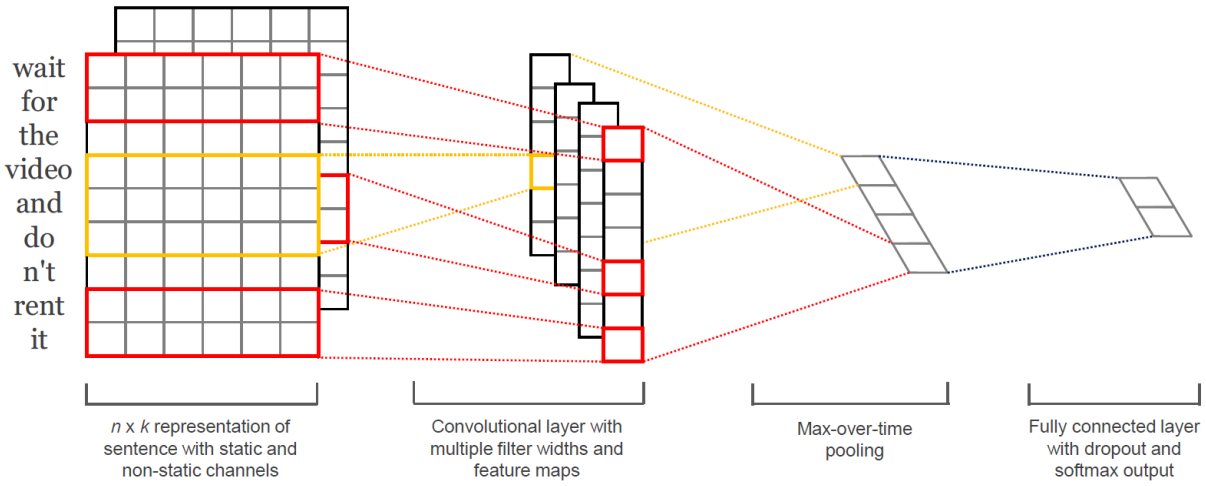


FIGURE C6. CNN architecture [86].

C.2. LSTM and BiLSTM

LSTM is one of the types of RNN architectures, consisting of feedback connections that can process entire sequences of data, being explicitly designed to avoid the long-term dependency problem by additively changing the state of the cell rather than transforming it completely. A LSTM cell is shown in Figure C7.

BiLSTM is an improved adaptation, more robust and sophisticated of LSTM, encoding information in both forward and backward direction. A brief illustration of BiLSTM architecture can be seen in Figure C8.

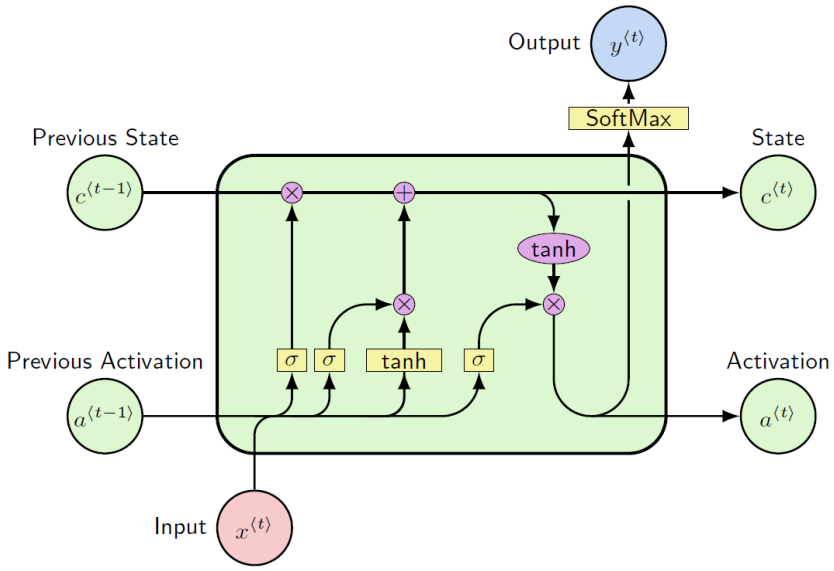


FIGURE C7. LSTM cell [102].

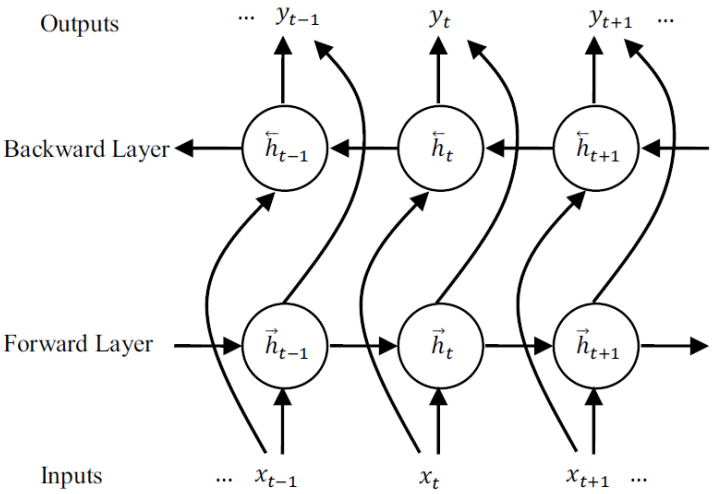


FIGURE C8. BiLSTM architecture [103].

D. Related Works

TABLE D1. Classical Approaches

Work	Pre-processing, Techniques, and Models	Results
[29, 2012]	**Techniques**: Emotion-based, profane words **Models**: Naive Bayes	**Twitter**: Accuracy: 0.673
[22, 2012]	**Techniques**: Gender information, personal pronouns, profane words, TF-IDF, weight of words **Models**: SVM	**MySpace**: Precision: 0.43 Recall: 0.16 F1-score: 0.23
[21, 2013]	**Pre-processing**: Stop words removal, Stemming **Techniques**: Content-based, user-based **Models**: SVM	**YouTube**: Precision: 0.77 Recall: 0.55 F1-score: 0.64
[10, 2013]	**Pre-processing**: Lowercase conversion, stop words removal **Techniques**: Session-based, personal pronouns, profane words, TF-IDF **Models**: Ensemble models	**Twitter, MySpace, Kongregate, Slashdot**: F1-score: 0.4
[31, 2015]	**Pre-processing**: Removal of stop words, extra characters, hyperlinks, unwanted characters **Techniques**: Fuzzy rule-based, extraction of features such as noun, adjective, pronoun and statistics on word occurrence **Models**: Genetic Algorithms	**Formspring**: Recall: 0.87 F1-score: 0.92 Accuracy: 0.86 **MySpace**: Recall: 0.98 F1-score: 0.91 Accuracy: 0.87
[24, 2016]	**Techniques**: Social-based, emotion-based, syntactic features, semantic features **Models**: Growing Hierarchical Self-Organizing Map	**Formspring**: System with reasonable performance
[5, 2016]	**Techniques**: Emotion-based, 2000 key terms **Models**: Naive Bayes, LDA	**Twitter**: Precision: 0.706 Recall: 0.705 F1-score: 0.704
[33, 2016]	**Pre-processing**: Removal of texts without context and duplicates **Models**: Participant-Vocabulary Consistency	**Twitter, ASK.fm**: Good compromise between Recall and Precision

[23, 2016]	**Pre-processing**: Stop words removal **Techniques**: Features such as network, activity, user and tweet contents, feature extraction to detect name, gender and age **Models**: Naive Bayes, Random Forest	**Twitter**: Unspecified
[30, 2016]	**Pre-processing**: Lowercase conversion, encoding cleaning, emoticons or emojis to expressions, spell correction, removal of very large and noisy texts, abbreviations to corrected words and removal of texts with more than 1000 words or less than 3 words **Models**: LSI	**Formspring**: Precision: 0.55
[34, 2017]	**Techniques**: Supervised Feature Selection **Models**: Extreme Learning Machine	**Twitter**: Supervised Feature Selection improved the results
[35, 2017]	**Pre-processing**: Lowercase conversion, Stemming, Tokenization, stop words removal **Techniques**: N-grams **Models**: SVM, Naive Bayes	**Formspring**: Accuracy: 0.971
[32, 2018]	**Pre-processing**: HTML tag removal, repeat more than twice words with characters that are normalized to two repetitions **Techniques**: Fuzzy FingerPrints **Models**: Fuzzy FingerPrints	**Formspring**: Precision: 0.355 Recall: 0.597 F1-score: 0.425
[26, 2018]	**Pre-processing**: Lemmatization, Tokenization, POS tagging **Techniques**: Subjectivity lexicons, N-grams **Models**: SVM	**ASK.fm**: F1-score: 0.643
[28, 2019]	**Pre-processing**: Lowercase conversion, Tokenization, stop words removal, encoding cleaning, spell correction, removal of very large and noisy texts **Techniques**: Emotion-based, TF-IDF, N-grams **Models**: SVM, NN	**ASK.fm**: F1-score: 0.919
[27, 2019]	**Techniques**: Emotion-based, textual-based, Medical Research Council, personality trait features **Models**: SVM, Logistic Regression, Random Forest	**Formspring**: F1-score: 0.45

TABLE D2. Deep Learning Approaches

Work	Pre-processing, Techniques, and Models	Results
[36, 2016]	**Pre-processing**: Hashtags removal, hyperlinks removal, elongated words treatment, accented characters replacement, non alphanumeric removal **Techniques**: Cost Function Adjusting, Threshold-Moving **Models**: CNN	**Formspring**: Precision: 0.540 Recall: 0.606 F1-score: 0.571 **Twitter**: Precision: 0.991 Recall: 0.975 F1-score: 0.983
[37, 2017]	**Pre-processing**: Text truncation for fixed size of 1014 tokens **Models**: CNN	**Multiple datasets**: Max-pooling performs better than other pooling types and the depth improves performance
[38, 2017]	**Pre-processing**: Tokenization, user mentions replacement, URLs replacement, unigrams and bigrams **Techniques**: Word Embeddings **Models**: Semantic-enhanced Marginalized Stacked Denoising Autoencoders	**Twitter, MySpace**: This model achieved remarkable performance
[40, 2018]	**Pre-processing**: Hyperlinks removal **Techniques**: Word Embeddings (GloVe) **Models**: CNN	**Twitter**: Precision: 0.77 Recall: 0.82 Accuracy: 0.95
[39, 2018]	**Pre-processing**: Lowercase conversion, hyperlinks removal, Tokenization, emoticons or emojis to expressions, user mentions replacement, hashtags replacement, non alphanumeric removal **Models**: CNN	**Twitter, YouTube, Facebook, Instagram, Tumblr, Gmail**: F1-score: 0.597
[46, 2018]	**Pre-processing**: Lowercase conversion, punctuation removal, Stemming, hashtag replacement, tokens removal with a document frequency less than 5 **Techniques**: Word Embeddings (Skip-gram Word2Vec) **Models**: CNN-LSTM	**Twitter, YouTube**: Outperformed other models

[43, 2018]	**Pre-processing**: Stop words removal, punctuation removal **Techniques**: Word Embeddings (GloVe, Sentiment-Specific Word Embeddings (SSWE)), others Transfer Learning approaches **Models**: CNN, LSTM, BiLSTM, BiLSTM with Attention	**Formspring, Twitter, YouTube, Wikipedia**: The best results were achieved with BiLSTM, BiLSTM with attention using GloVe and Model Level Transfer Learning
[44, 2018]	**Pre-processing**: Lowercase conversion, stop words removal, punctuation removal **Techniques**: Word Embeddings (GloVe, SSWE), others Transfer Learning approaches, oversampling **Models**: CNN, LSTM, BiLSTM, BiLSTM with Attention	**Formspring, Twitter, Wikipedia**: Oversampling significantly improved the performance of all models
[42, 2018]	**Pre-processing**: Hyperlinks removal, elongated words treatment, encoding and unicodes removal **Techniques**: Word Embeddings (GloVe, Word2vec) **Models**: CNN, CNN-LSTM	**Formspring**: F1-score: 0.444
[9, 2019]	**Pre-processing**: Lowercase conversion, Lemmatization, elongated words treatment **Techniques**: Word Embeddings (GloVe, Word2vec, ELMo), ADASYN **Models**: LSTM, BiLSTM, GRU	**Formspring**: BiLSTM using ELMo outperformed all other models
[49, 2019]	**Pre-processing**: Lowercase conversion, hashtags removal, username mentions removal, hyperlinks removal, Lemmatization, elongated words treatment, spell correction, expansion all apostrophes containing words **Techniques**: Word Embeddings (GloVe Common Crawl), SMOTE, Class Weights **Models**: CNN, LSTM, BiLSTM, CNN-LSTM, GRU, BiGRU, CNN-GRU, CNN-BiGRU, LSTM-CNN, BiLSTM-CNN, GRU-CNN, BiGRU-CNN	OLID: BiLSTM-CNN using GloVe Common Crawl was the best model, trained in 5 epochs and no dropout layers

[47, 2019]	**Pre-processing**: Lowercase conversion, punctuation removal, Tokenization, numbers replacement, extra white spaces removal **Techniques**: Word Embeddings (fastText) **Models**: CNN, BiGRU-CNN, BiGRU-BiLSTM-CNN	**Vietnamese Language and Speech Processing 2019 challenge: Hate Speech Detection on Social Networks**: BiGRU-BiLSTM-CNN achieved the best performance
[57, 2019]	**Pre-processing**: Lowercase conversion, emoticons or emojis to expressions, hashtag segmentation, URLs replacement, consecutive user mention limited to three times **Techniques**: Contextualized Word Embeddings **Models**: LSTM, BERT	OLID: BERT was the model that achieved the best result
[48, 2019]	**Pre-processing**: Lowercase conversion, punctuation removal, hashtag segmentation, elongated words treatment, encoding and unicodes removal, user mentions replacement, URLs replacement, numbers replacement, hashtags replacement **Techniques**: Contextualized Word Embeddings **Models**: BERT-CNN, BERT-LSTM	**Twitter**: BERT-CNN outperformed previous works
[53, 2019]	**Pre-processing**: Hashtag segmentation and emoji substitution **Techniques**: Contextualized Word Embeddings **Models**: BERT	OLID: Macro average F1-score: 0.829
[45, 2020]	**Pre-processing**: Lowercase conversion, punctuation removal, emoticons or emojis to expressions, hashtag segmentation, spell correction, expansion all apostrophes containing words **Techniques**: Word Embeddings (GloVe, Word2vec, fastText) **Models**: Character-CNN, LSTM, BiLSTM	**Twitter, Facebook**: LSTM and BiLSTM were the best models
[58, 2020]	**Pre-processing**: Hashtag segmentation, user mentions replacement, rare words substitution, truncation to max size of 64 containing words **Techniques**: Contextualized Word Embeddings, MTL **Models**: BERT	OLID: F1-score: 0.8382 SOLID: F1-score: 0.9151

[55, 2020]	**Pre-processing**: Conversion of emojis into word representations, hashtag segmentation, abbreviation expansion, bad words replacement, spell correction, lowercase conversion, Stemming, Lemmatization, removal of user mentions, URLs, hashtags, emojis, e-mails, dates, numbers, punctuation, consecutive character repetitions, offensive words and stop words **Techniques**: Contextualized Word Embeddings **Models**: ALBERT	SOLID: Macro average F1-score: 0.9204

E. Deep Learning Model Architectures

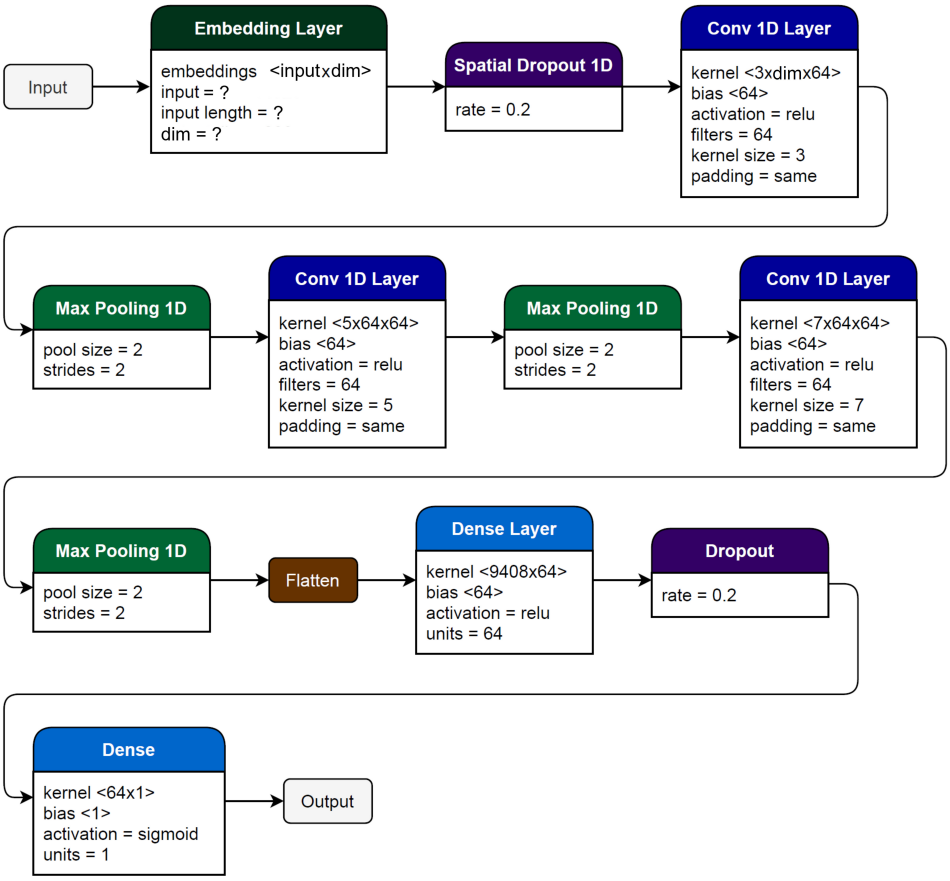


FIGURE E1. CNN architecture

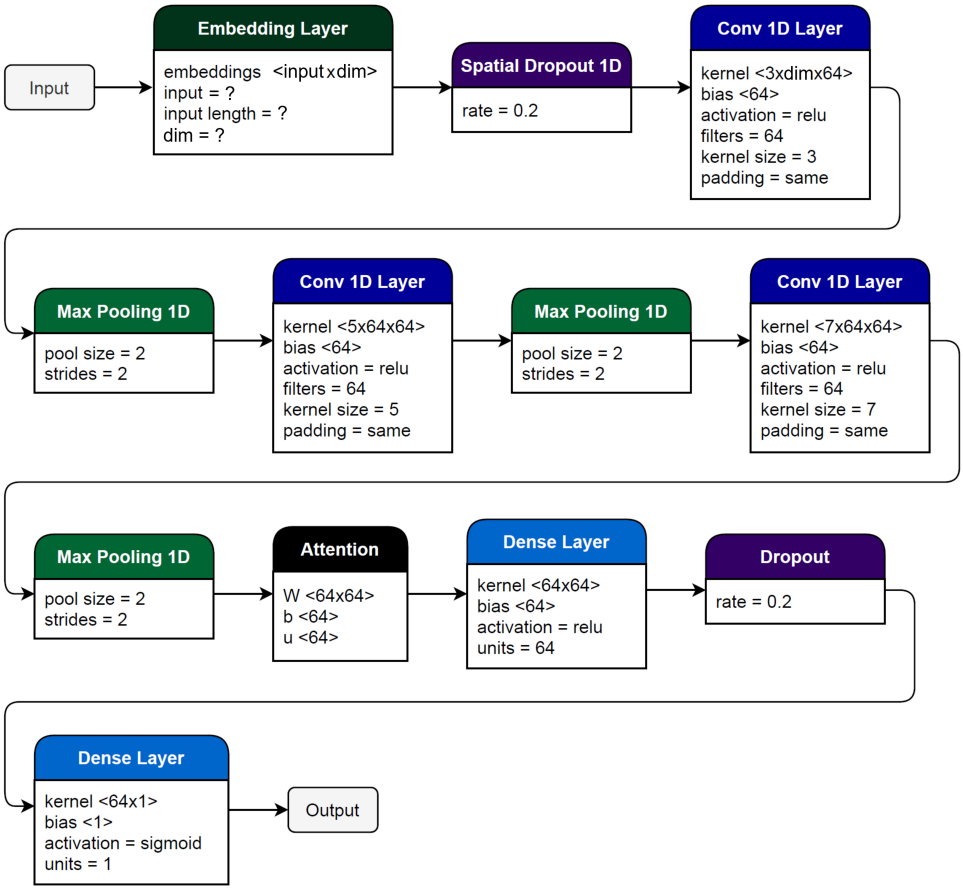


FIGURE E2. CNN-Attention architecture

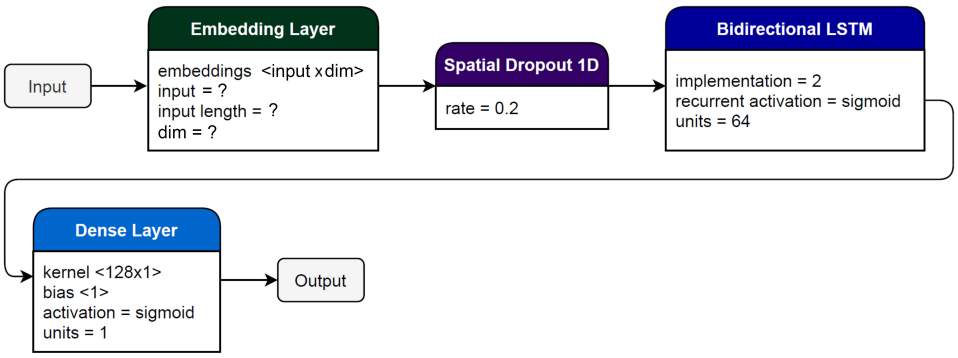


FIGURE E3. BiLSTM architecture

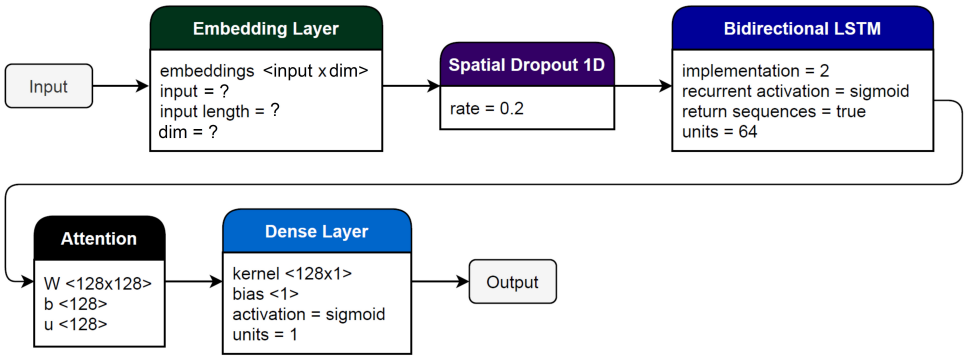


FIGURE E4. BiLSTM-Attention architecture

F. Deep Learning - Training History

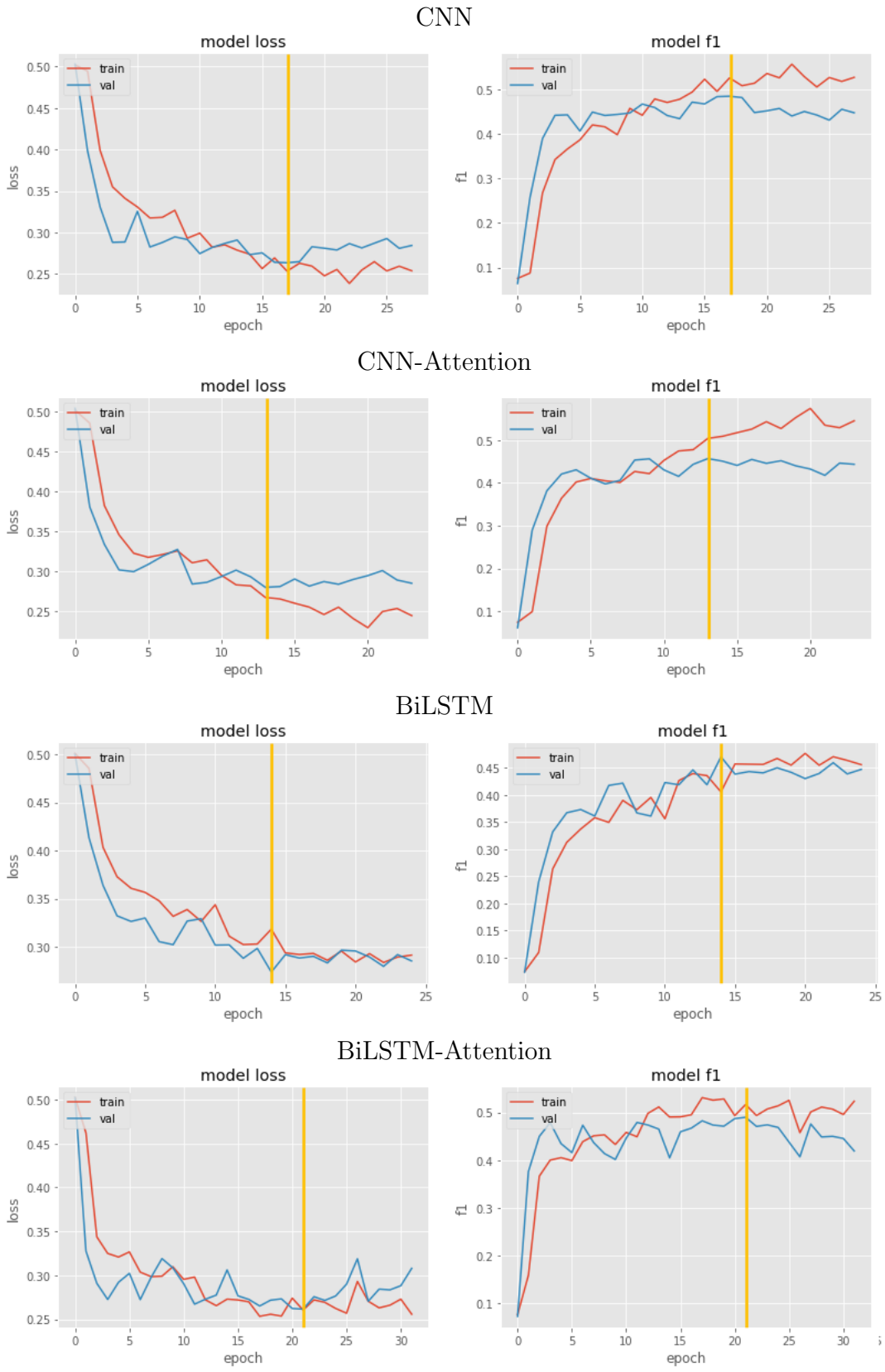


FIGURE F1. Training history, using GloVe Twitter and Macro Soft-F1 Loss for the Formspring dataset

FIGURE F2. Training history, using GloVe Common Crawl and Macro Soft-F1 Loss for the Formspring dataset

FIGURE F3. Training history, using GloVe Twitter and Macro Soft-F2 Loss fo the Formspring dataset

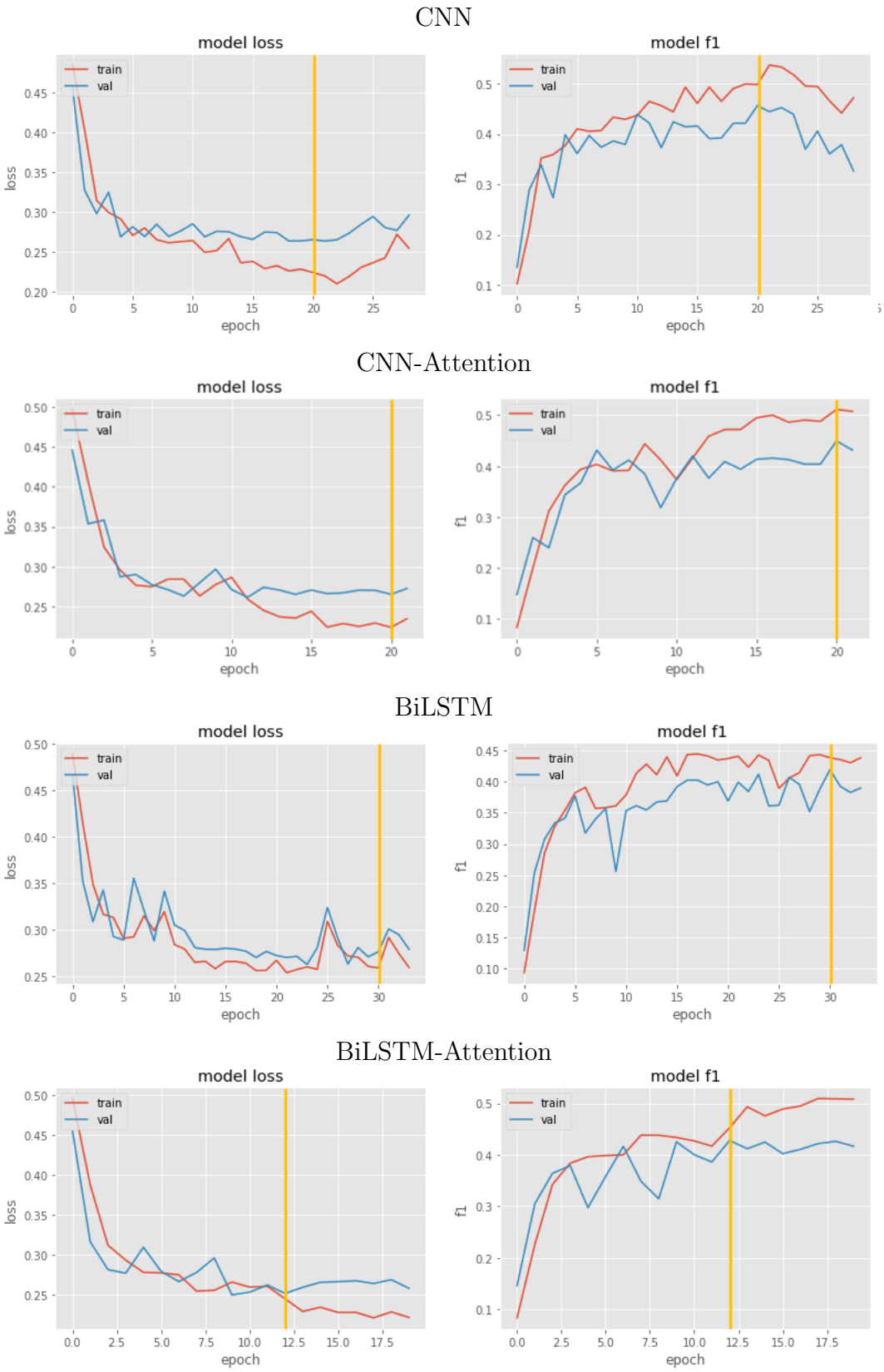


FIGURE F4. Training history, using GloVe Common Crawl and Macro Soft-F2 Loss for the Formspring dataset

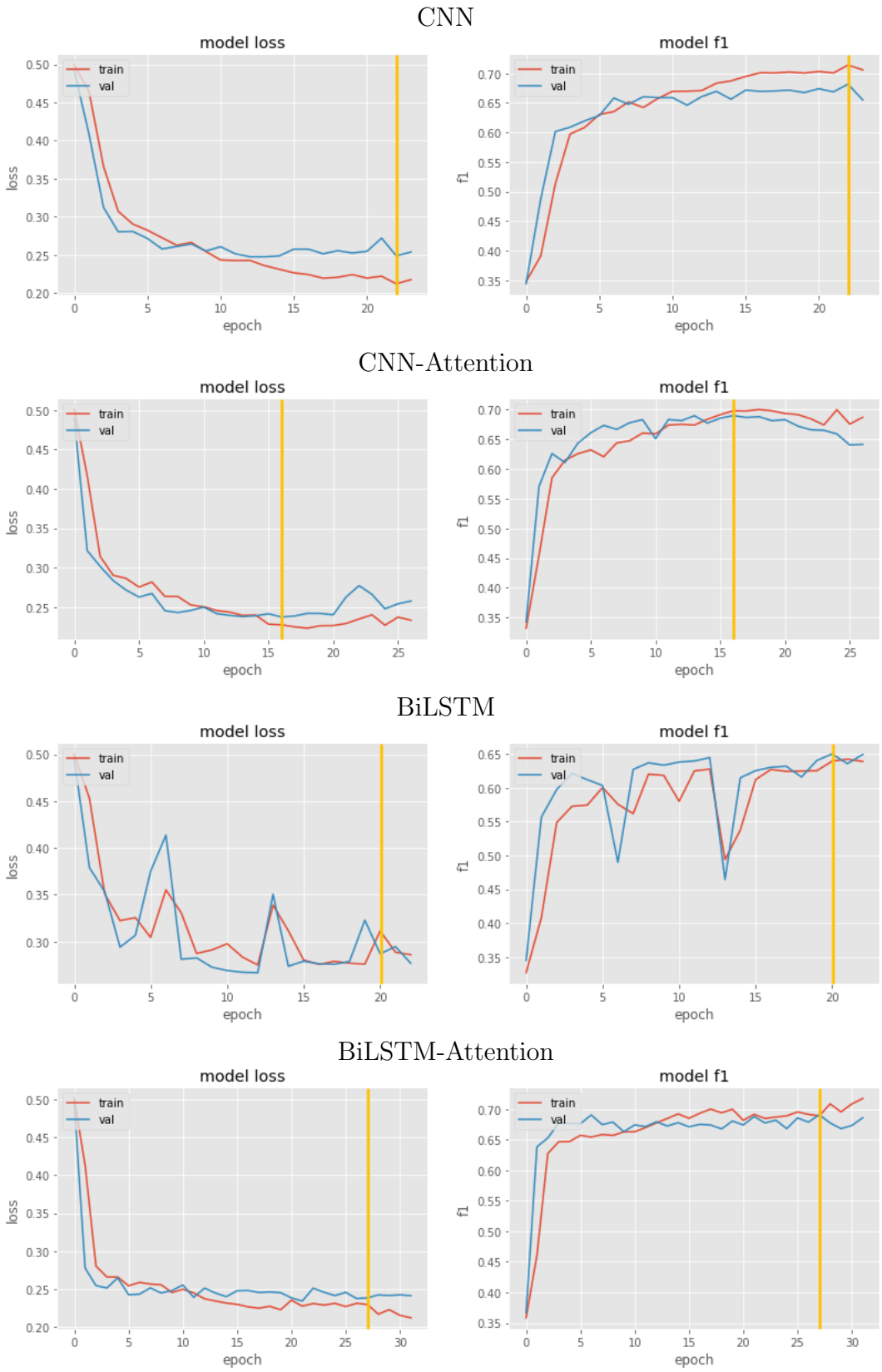


FIGURE F5. Training history, using GloVe Twitter and Macro Soft-F1 Loss for the OLID

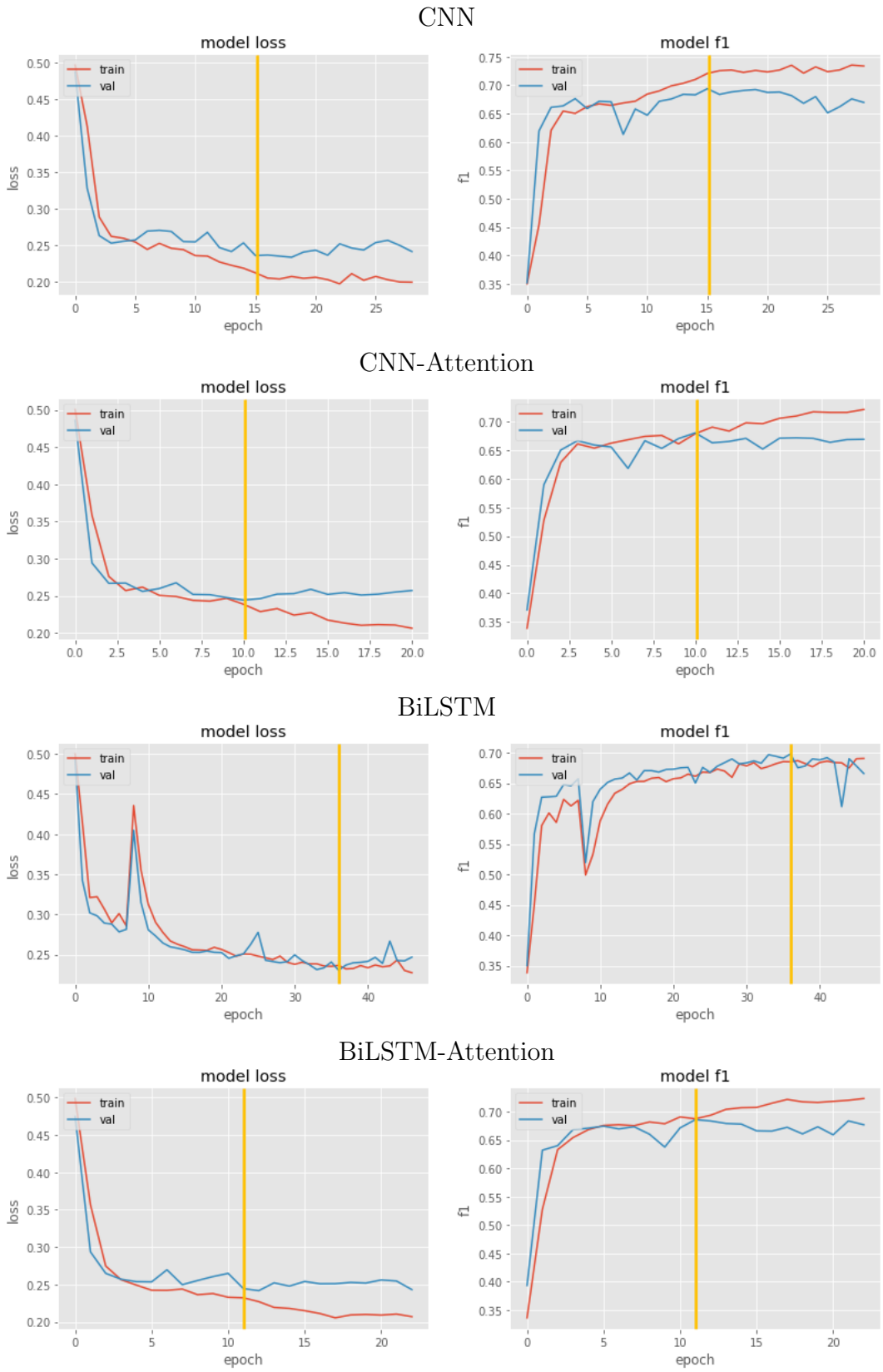


FIGURE F6. Training history, using GloVe Common Crawl and Macro Soft-F1 Loss for the OLID dataset

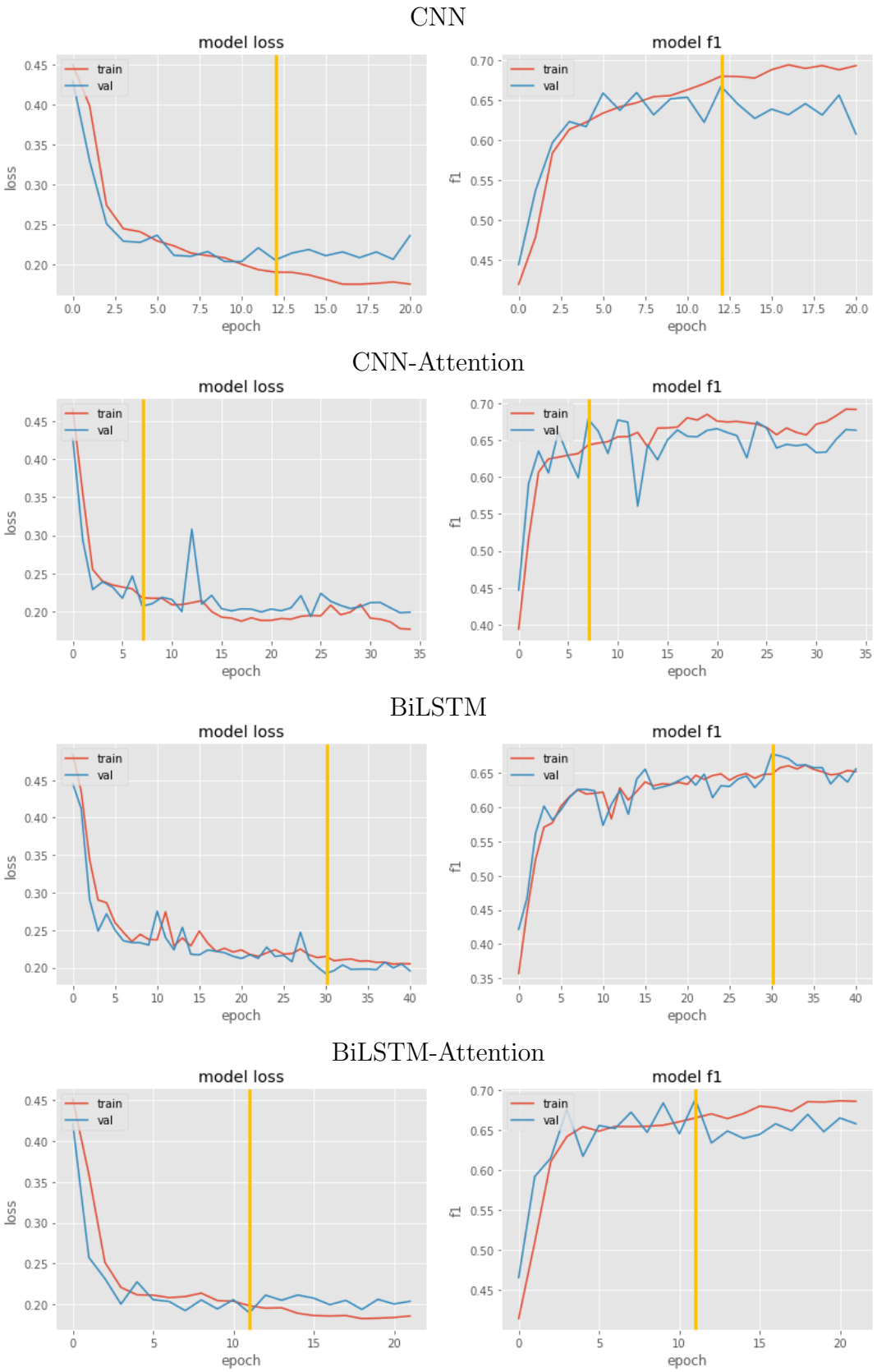


FIGURE F7. Training history, using GloVe Twitter and Macro Soft-F2 Loss for the OLID dataset

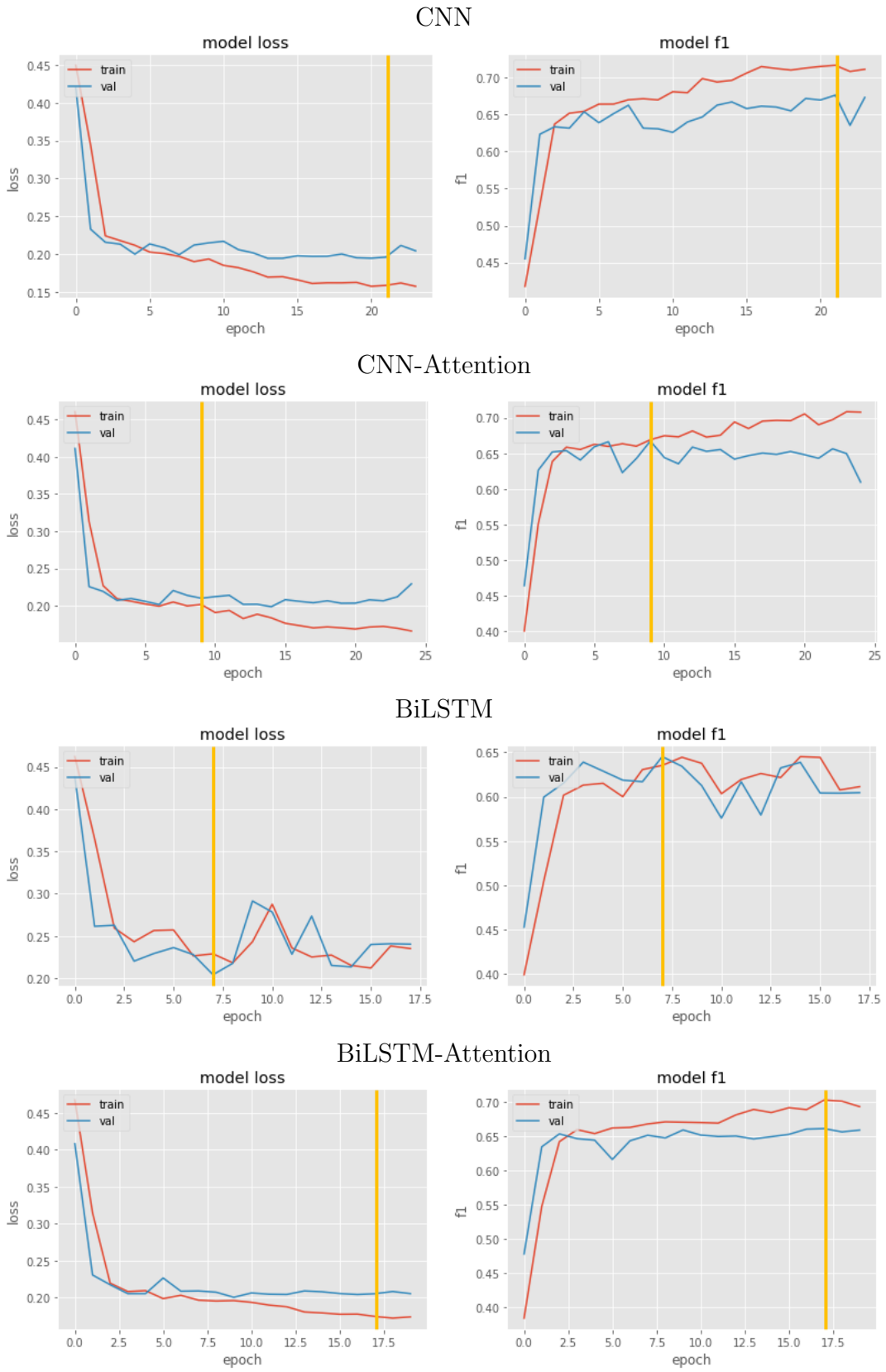
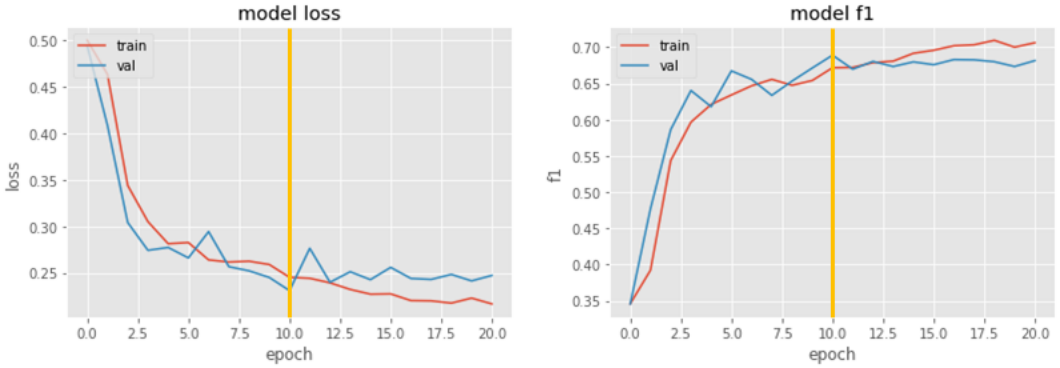
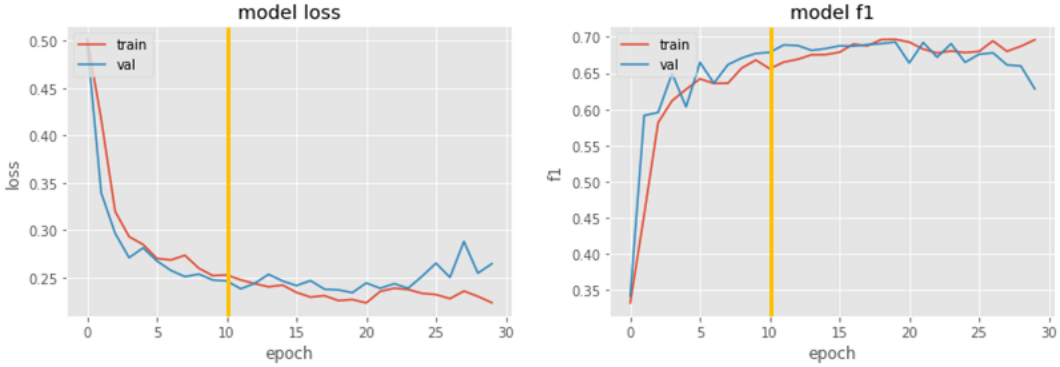


FIGURE F8. Training history, using GloVe Common Crawl and Macro Soft-F2 Loss for the OLID dataset

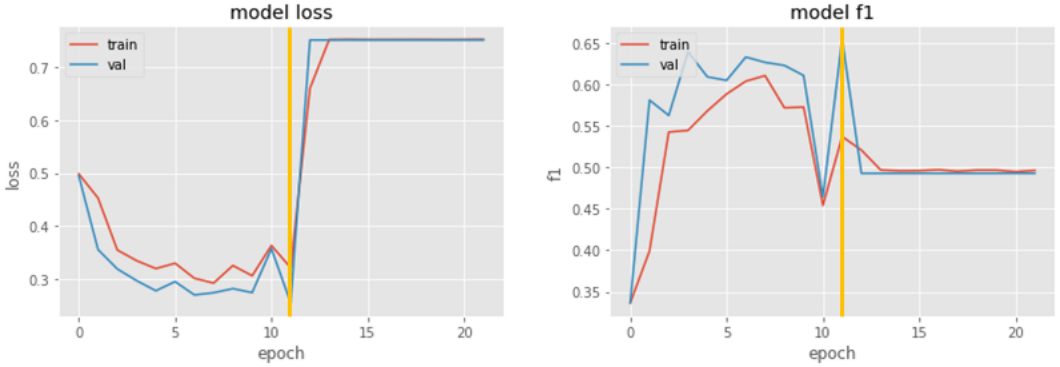
CNN



CNN-Attention



BiLSTM



BiLSTM-Attention

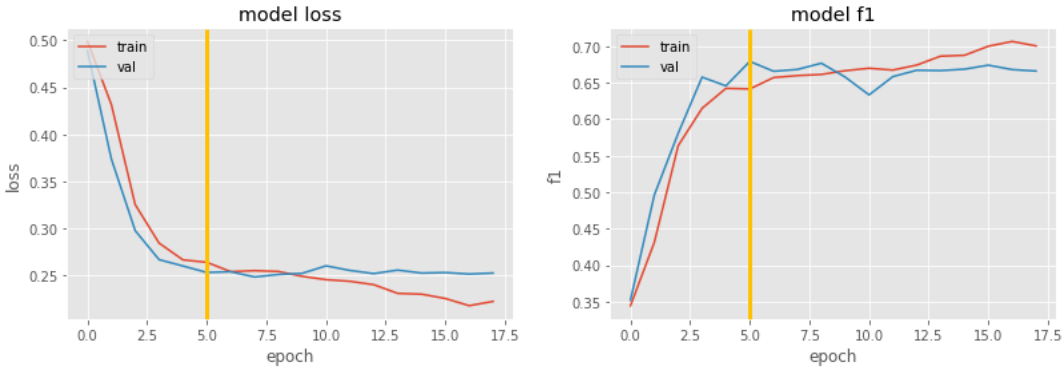
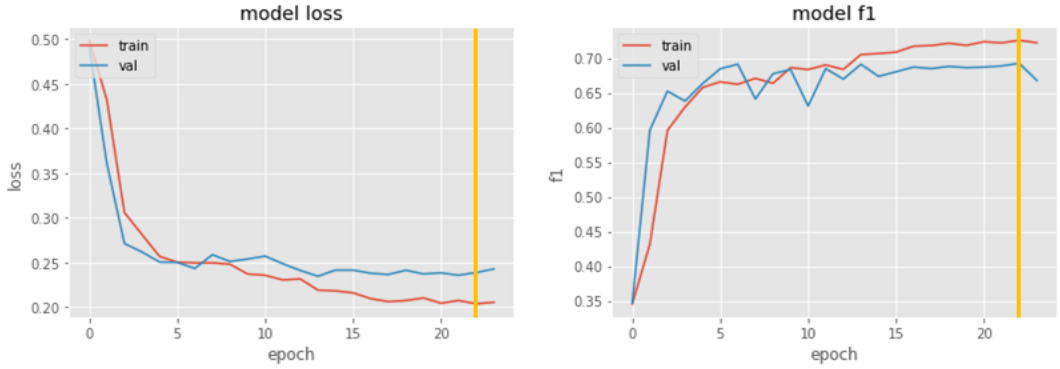
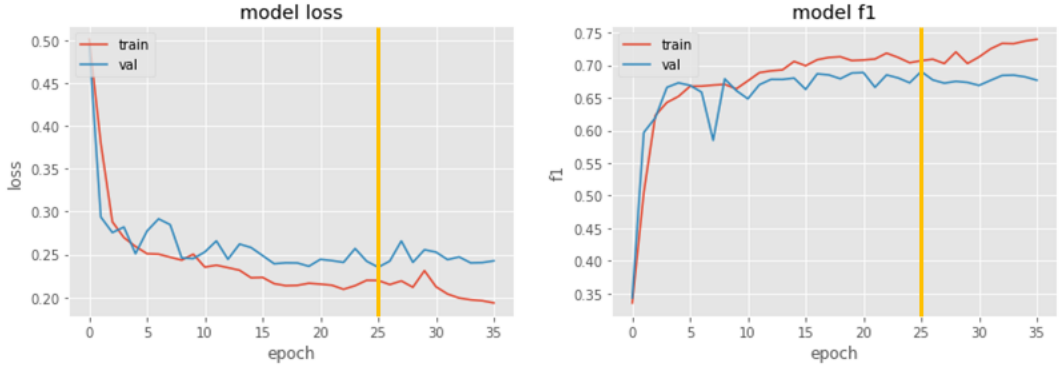


FIGURE F9. Training history, using GloVe Twitter and Macro Soft-F1 Loss for the SOLID

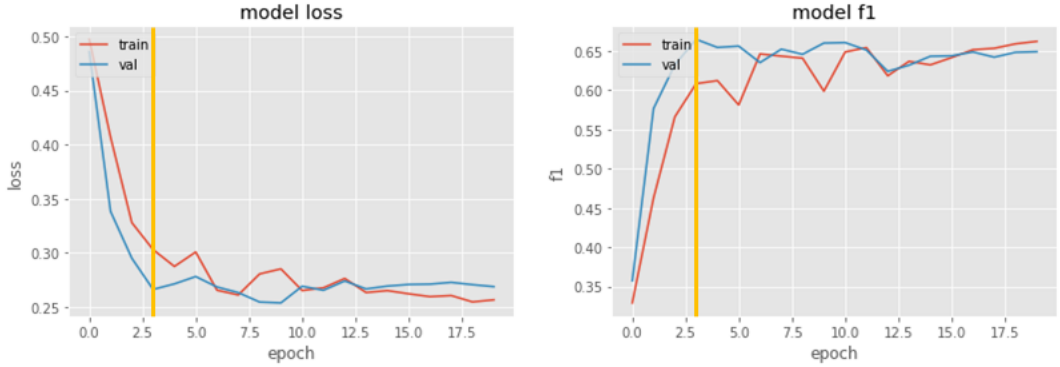
CNN



CNN-Attention



BiLSTM



BiLSTM-Attention

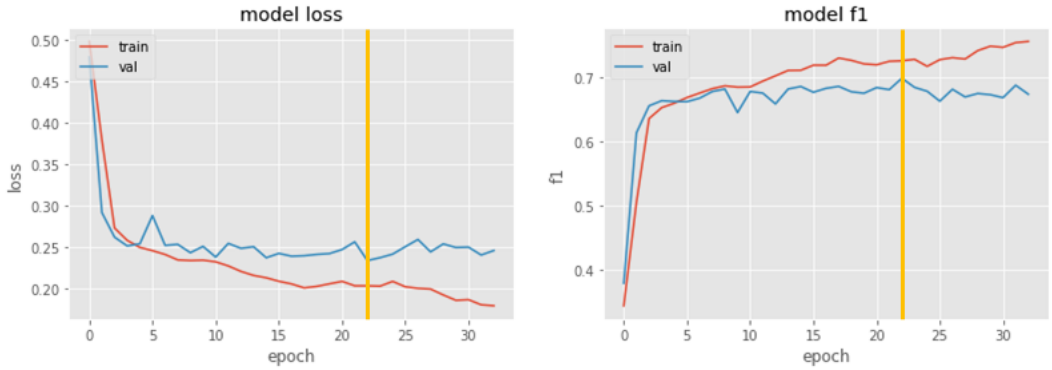


FIGURE F10. Training history, using GloVe Common Crawl and Macro Soft-F1 Loss for the SOLID

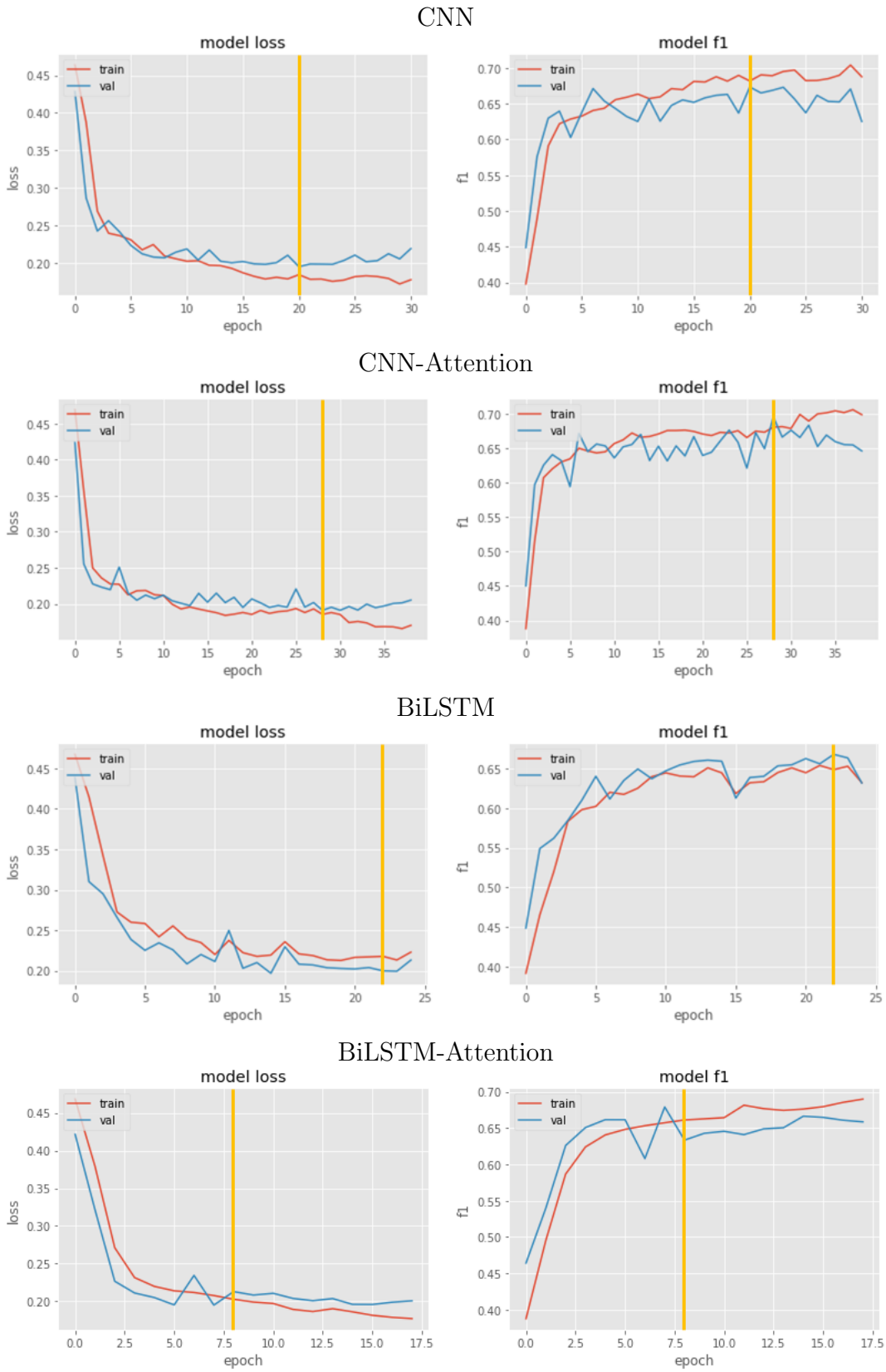
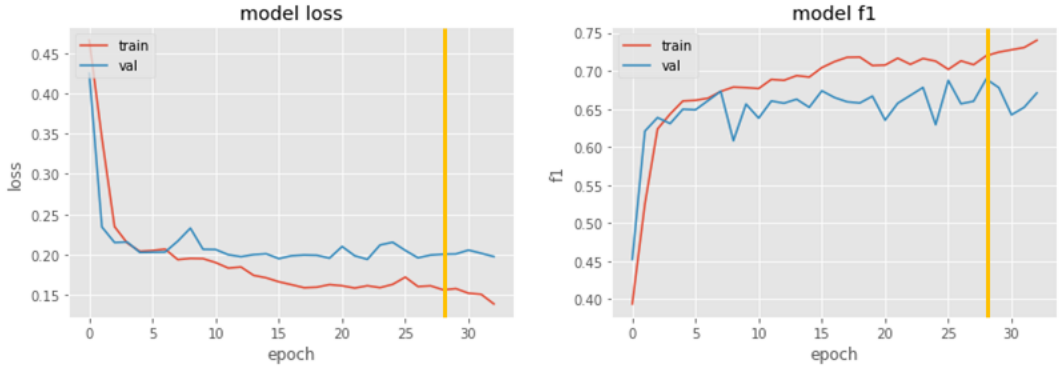
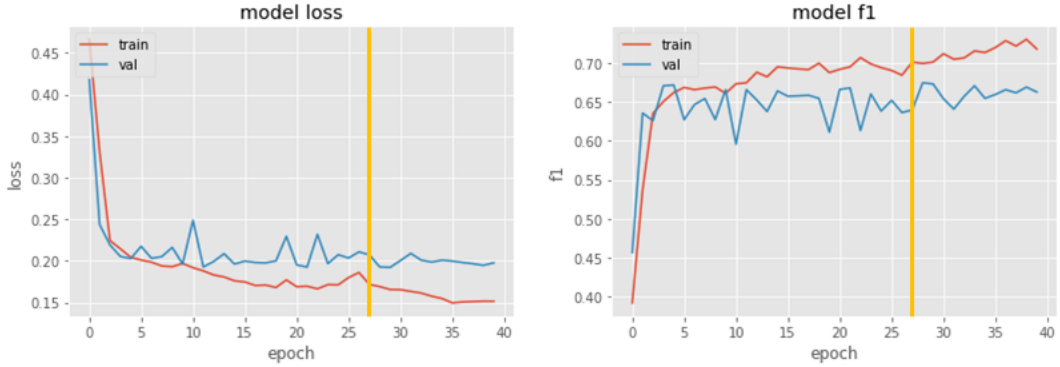


FIGURE F11. Training history, using GloVe Twitter and Macro Soft-F2 Loss for the SOLID

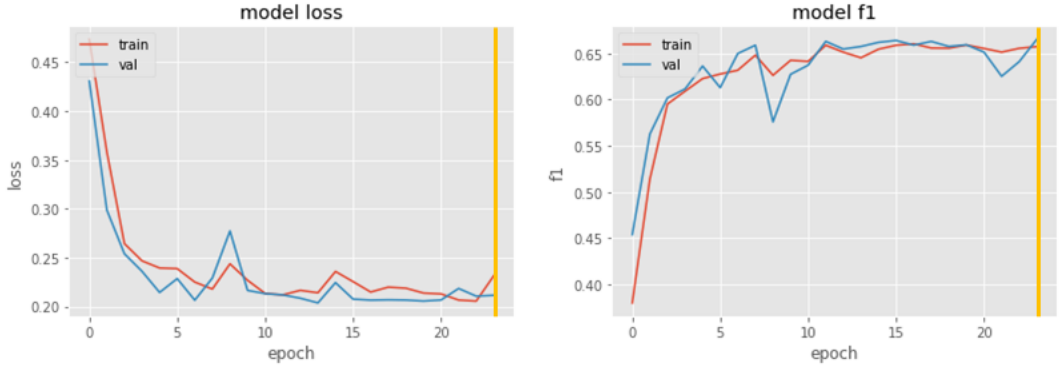
CNN



CNN-Attention



BiLSTM



BiLSTM-Attention

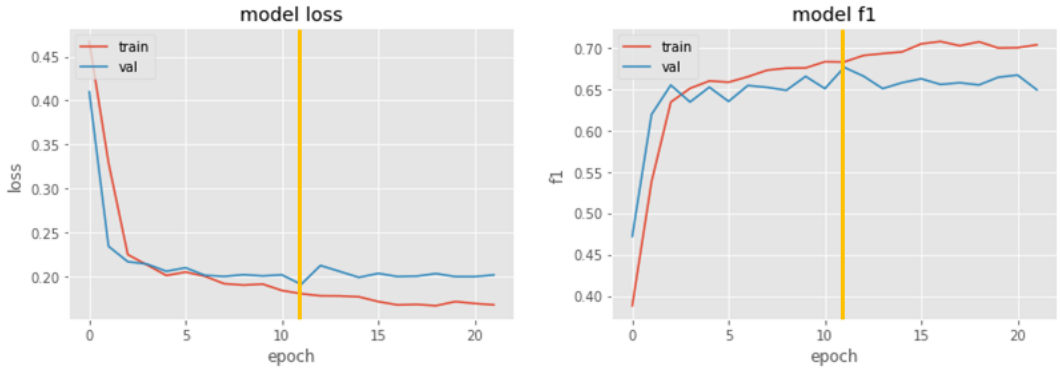


FIGURE F12. Training history, using GloVe Common Crawl and Macro Soft-F2 Loss for the SOLID