NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

LANGUAGE TECHNOLOGY

MASTER'S THESIS

Offensive Language Detection in Tweets Using Machine Learning Methods

Christina G. Christodoulou

ATHENS

SEPTEMBER 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός Προσβλητικής Γλώσσας Σε Tweets Χρησιμοποιώντας Μεθόδους Μηχανικής Μάθησης

Χριστίνα Γ. Χριστοδούλου

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2022

MASTER'S THESIS

Offensive Language Detection in Tweets Using Machine Learning Methods

Christina G. Christodoulou

A.M.: LT1200027

SUPERVISOR: Panagakis Ioannis, Associate Professor U.O.A

EXAMINATION COMMITTEE:

Panagakis Ioannis, Associate Professor U.O.A

Katsouros Vassilis, Research Director, ILSP/Athena Research Center

Ioannakis Georgios Alexis, Researcher C', ILSP/Athena

September 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός Προσβλητικής Γλώσσας Σε Tweets Χρησιμοποιώντας Μεθόδους Μηχανικής Μάθησης

Χριστοδούλου Γ. Χριστίνα

Α.Μ.: LT1200027

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Παναγάκης Ιωάννης, Αναπληρωτής Καθηγητής Ε.Κ.Π.Α

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Παναγάκης Ιωάννης, Αναπληρωτής Καθηγητής Ε.Κ.Π.Α

Κατσούρος Βασίλης, Διευθυντής Ερευνών, ΙΕΛ/ΕΚ Αθηνά

Ιωαννάκης Γεώργιος Αλέξης, Ερευνητής Γ', ΙΕΛ/ΕΚ Αθηνά

Σεπτέμβριος 2022

ABSTRACT

Undoubtedly, offensive language has become ubiquitous in social media over the last years due to the increasing popularity of social media platforms. The growing number of users that tend to post offensive content targeting individuals or groups has led to significant repercussions not only for the well-being of the targets, but also for society itself. This has raised concern in governments, social media companies as well as academic and social communities, who have made concerted efforts to curb the dissemination of offensive language online and create a safer digital space. Nevertheless, despite their endeavors, the need to rapidly process huge amounts of content in order to detect and report offensive language has made the development of machine learning systems more than imperative. Consequently, in the present thesis, three different machine learning models, which perform binary text classification, are introduced to detect offensive language in English texts from Twitter. The proposed models, which constitute two simple classifiers and a Bidirectional Stacked LSTM, utilize contextual embeddings pooled from $BERT_{LARGE\text{-}Uncased}$ by fine-tuning its various layers on four training datasets combined in one. The data preparation process involved data cleaning and preprocessing as well as data down-sampling to handle class imbalance. The effectiveness of the proposed methods is evaluated on two available test sets, OLID 2019 and OLID 2020, based on six metrics, the learning curves of loss and accuracy as well. Comparative analysis between those methods demonstrates that the concatenation of the last four hidden layers of BERT fed in a classifier outperforms the other models by achieving 77.8% and 86.8% Macro-F1 scores on the two test sets respectively. Comparison with previous related methods indicates that, although the results are satisfactory, there is room for further experimentation and improvement in the future.

SUBJECT AREA: Offensive Language Detection in Tweets Using Machine Learning Methods

KEYWORDS: Offensive Language Detection, Offensive Language, Twitter, Machine Learning, Deep Learning, Natural Language Processing, NLP, Text Classification, Social Media

ΠΕΡΙΛΗΨΗ

Αναμφίβολα, η προσβλητική γλώσσα έχει γίνει διαδεδομένη στα μέσα κοινωνικής δικτύωσης τα τελευταία χρόνια λόγω της αυξανόμενης δημοτικότητάς τους. Ο αυξανόμενος αριθμός χρηστών που τείνουν να δημοσιεύουν προσβλητικό περιεχόμενο στοχεύοντας σε άτομα ή ομάδες επιφέρει σοβαρές επιπτώσεις όχι μόνο στην ευημερία των ατόμων, αλλά και στην ίδια την κοινωνία. Το γεγονός αυτό έχει προκαλέσει ανησυχία στις κυβερνήσεις, στις εταιρείες μέσων κοινωνικής δικτύωσης, αλλά και στις ακαδημαϊκές και κοινωνικές κοινότητες, οι οποίες έχουν καταβάλει συντονισμένες προσπάθειες για τον περιορισμό διάδοσης της προσβλητικής γλώσσας στο διαδίκτυο και τη δημιουργία ενός ασφαλέστερου διαδικτυακού χώρου. Ωστόσο, παρά τις προσπάθειές τους, η ανάγκη ταχείας επεξεργασίας ογκώδους πληροφορίας για τον εντοπισμό και την αναφορά προσβλητικής γλώσσας έχει καταστήσει την ανάπτυξη συστημάτων μηχανικής μάθησης κάτι παραπάνω από επιτακτική. Συνεπώς, στην παρούσα διπλωματική εργασία, εισάγονται τρία διαφορετικά μοντέλα μηχανικής μάθησης, τα οποία εκτελούν δυαδική ταξινόμηση κειμένου, για τον εντοπισμό προσβλητικής γλώσσας σε αγγλικά δημοσιεύματα κειμένων από το Twitter. Τα προτεινόμενα μοντέλα, τα οποία αποτελούνται από δύο απλούς ταξινομητές και ένα Bidirectional Stacked LSTM, αξιοποιούν τα contextual embeddings που προέρχονται από το $\text{BERT}_{\text{LARGE-Uncased}}$ με fine-tuning του σε τέσσερα σύνολα δεδομένων εκπαίδευσης συγκεντρωμένα σε ένα. Η διαδικασία προετοιμασίας των δεδομένων περιλαμβάνει καθαρισμό και προ-επεξεργασία των δεδομένων, καθώς και υποδειγματοληψίας των δεδομένων για την αντιμετώπιση της ανισορροπίας των κλάσεων. Η αποτελεσματικότητα των προτεινόμενων μεθόδων αξιολογείται σε δύο διαθέσιμα σύνολα δεδομένων αξιολόγησης, τα OLID 2019 και OLID 2020, με βάση έξι μετρικές, καθώς και τις καμπύλες μάθησης της απώλειας και της ακρίβειας. Η συγκριτική ανάλυση μεταξύ αυτών των μεθόδων αποδεικνύει ότι η συνένωση των τεσσάρων τελευταίων κρυφών επιπέδων του BERT που περνούν σε έναν ταξινομητή υπερτερεί των άλλων μοντέλων επιτυγχάνοντας 77,8% και 86,8% Macro-F1 σκορ στα δύο σύνολα δεδομένων αξιολόγησης αντίστοιχα. Η σύγκριση με προηγούμενες συναφείς μεθόδους αποκαλύπτει ότι, μολονότι τα αποτελέσματα είναι ικανοποιητικά, υπάρχουν περιθώρια για περισσότερο πειραματισμό και βελτίωση στο μέλλον.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εντοπισμός Προσβλητικής Γλώσσας σε Tweets Χρησιμοποιώντας Μεθόδους Μηχανικής Μάθησης

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Εντοπισμός Προσβλητικής Γλώσσας, Προσβλητική Γλώσσα, Twitter, Μηχανική Μάθηση, Βαθιά Μάθηση, Επεξεργασία Φυσικής Γλώσσας, Ταξινόμηση κειμένου, Κοινωνικά Δίκτυα, Μέσα Κοινωνικής Δικτύωσης

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation to my supervisor Ass. Professor Panagakis and Dr. Ioannakis, for their insightful comments and suggestions. This endeavor would not have been possible without their consistent support and belief in me. I would also like to extend my sincere thanks to the researchers that created the datasets used for the training and evaluation of the machine learning models, the researchers that offered available code, which was followed as a guide, as well as the creators of the provided figures. Last but not least, I could not have undertaken this journey without my family and closest friends, who have always been by my side in every way.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SOURCE CODES

1. Data and the complete source code of the thesis are available at: https://github.com/christinacdl/Thesis_Detection_of_Offensive_Language
2. *Utilizing Transformer Representations Efficiently* on Kaggle is available at: https://www.kaggle.com/code/rhtsingh/utilizing-transformer-representations-efficiently
3. The code used for the BERT Bi-Stacked-LSTM from the article *Utilizing BERT Intermediate Layers for Aspect Based Sentiment Analysis and Natural Language Inference* (Song et al., 2020) is available at: GitHub - avinashsai/BERT-Aspect: BERT Fine-tuning for Aspect Based Sentiment Analysis

1. INTRODUCTION

1.1 Overview

Without a doubt, social networking has become an integral part of our lives due to its prevalence in the recent years. Mass communication, the expression of one's self as well as the rapid dissemination of information have been facilitated by social media platforms, such as Twitter and Facebook, considering their convenient and public free access. Moreover, social media possess the power to promote social movements worldwide through hashtag activism. Hashtags can go instantly viral and communicate the proper message. Certain memorable hashtags include *#MeToo* and *#BlackLivesMatter*, which gather advocates and victims of sexual assault and harassment, racial inequality and police violence respectively (Sossi, 2022).

Although social media networks offer many benefits, the anonymity and invisibility combined with online disinhibition have resulted in the demonstration of negative behavior online (Wright et al., 2019). In other words, there is an increasing number of users who tend to engage online and feel comfortable enough to unleash their negative sentiment towards others through the creation of offensive content without taking into account the consequences. Offensive content has become ubiquitous and can take many forms of media, for instance text, images, videos and audio. These forms can sometimes be combined (Stop Hate UK). Offensive content in the type of text may constitute derogatory, racist, sexist, xenophobic statements, insults, threats as well as pejorative terms and slurs towards individuals or groups (Stop Hate UK; Pitenis et al., 2020).

Constant use and exposure to social media may have a detrimental effect on the mental health, the online experience and even cause self-harm of the targeted individuals or groups, not to mention the negative impact on the community. To illustrate, a 19-year-old woman, named Phoebe Jameson, reported that she experienced online abuse every day during 2020 (Baggs, 2021), whereas Phoebe Prince, a 15-year-old student, committed suicide by hanging after being harassed by other students through texts on Facebook (Goldam, 2010). According to a pilot study conducted among 130 high school students, 33% of the participants had experienced cyberbullying via social media networks. It is also worth-mentioning that almost one in two cyber victims selected to delete their social media accounts (Oblad, 2019). In the US, there has been a dramatic rise in the use of abusive and hateful language towards pro-choice abortion activists on Twitter concerning the withdrawal of the constitutional right to abortion voted by the Supreme Court (Amnesty International UK, 2022). An additional report has demonstrated a spike in the use of online offensive language in the UK and the US by 20% during the Covid-19 pandemic and lockdown (Baggs, 2021).

The propagation of offensive language and the hate-based incidents through social media have raised concern in governments, communities and the social media platforms, who have come to realize that detecting and dealing with illegal and offensive content while protecting freedom of expression are of paramount importance. In fact, on 25 March 2022 the European Parliament and the European Union came to a political agreement regarding a piece of legislation named the Digital Services Act (DSA), which focuses on creating a safer digital

space for the protection of the users' fundamental rights (European Commission, 2022). In this way, social media companies are required to monitor and eradicate any illicit and harmful content on their platforms by following certain procedures and applying safety policies (Modinos, 2022). This is in line with the UK's initiative to introduce a draft Online Safety Bill on May 2021, which is currently under scrutiny and planning to be implemented (GOV.UK, 2022; Wakefield, 2021). The Internet intermediaries, such as Microsoft, YouTube, Twitter and Facebook, have declared intentions on their part to detect and immediately eliminate the offensive and illegal content from their platforms in order to protect their users and freedom of expression (BBC, 2016). Several companies are constantly updating their policies so as to alleviate the harmful or offensive content from their platforms, which vary from the appearance of warning prompts to the suspension of accounts. According to Twitter, the new prompt feature *Want to review this before Tweeting?,* which makes users reconsider their posts in the event of potential offensive language detection, has revealed that users were less likely to send or receive offensive messages when prompted (Hern, 2021). Nevertheless, even actions like the suspension of accounts cannot catch up with the propagation of violence and terrorism online. It is for this reason that social media companies have been denounced for their limited and belated attempts to detect and deal effectively with such phenomena (Cain & Gonzalez, 2017).

The mitigation of hateful user-generated content has been conducted manually by online content moderators, who are employed by social media companies to deem the appropriateness of the content according to each company's internal policies by maintaining or deleting it. However, the manual social media moderation is a time-consuming, laborious and dependent on human judgement process (Pitsilis et al., 2018). Furthermore, it has been revealed that online content moderators suffer from various mental health problems, for instance Post-Traumatic Stress Disorder (PTSD) and depression, due to their continuous exposure to offensive, violent and pornographic content (Arsht & Etcovitch, 2018). The challenges and the severe consequences of manual social media moderation along with the increase of social media content have shown that an automated process is imperative.

The automatic detection of offensive content in the form of written language in social media has become a significant researched task in the Natural Language Processing (NLP) field over the last years, since it has the potential to curtail online content rapidly and efficiently, not to mention reducing the workload of online content moderators (Chaudhary et al., 2021). It has been mainly approached as a binary or multi-class text classification task, which utilizes annotated data from one or more social media platforms as training input for machine learning algorithms. Supervised learning is applied for classification tasks, as the machine learning models learn a mapping between input and target variables in order to predict results based on new and unseen data. Various supervised machine learning algorithms have been utilized for offensive language identification ranging from traditional models, like Naïve Bayes, Support Vector Machines (SVM), Logistic Regression, to deep learning models, like Recurrent Neural Networks (RNNs) and Transformers. Taking into consideration the advances in technology, recent related work has begun employing state-of-the-art deep learning models, as they are able to automatically learn features from the input data as well as to deploy large amount of computation and data, hence offering remarkable results (Young

et al., 2018; Mutanga et al., 2020).

Despite the fact that a lot of research has primarily focused on the English language owing to the unlimited available linguistic resources from the so-called *lingua franca*, a notable number of late studies have started to bridge the gap in this area by shifting their attention to other languages as well, for instance Spanish (Aragón et al., 2018), Indonesian (Ibrohim et al., 2019) and Greek (Pitenis et al., 2020). Among the social media platforms, Twitter has been researched the most in terms of offensive language due to free access to a great number of posts through the Twitter API and the already collected data from previous studies.

The present thesis intends to contribute to offensive language detection in the English language focusing on Twitter by utilizing state-of-the-art machine learning methods. Deep learning models, namely LSTM Networks and BERT, the pre-trained Transformer model, are developed for a binary text classification task that aims to discriminate between offensive and not offensive Twitter posts. Consequently, this study constitutes an endeavor towards curbing the dissemination of offensive content on social media through automatic detection. The thesis provides a structured and intuitive approach concerning data preparation, model training and evaluation in Python programming language, comparative analysis of models as well as overviews outlining conclusions, limitations and future work. The key contributions of the proposed study are as follows:

- Analyzing the procedure followed regarding data wrangling, resampling and preprocessing of four different datasets in order to handle class imbalance and create appropriate training, validation and test inputs for BERT.
- Fine-tuning $BERT_{LARGE\text{-}Uncased}$ while following the feature-based approach by retrieving different Transformer contextual embeddings through the development of three proposed model architectures:
    1. A custom classifier model by pooling the [CLS] token embeddings from the last hidden layer of BERT and then feeding them into a linear layer that classifies the outputs.
    2. A custom classifier model by concatenating the token representations from the last four hidden layers of BERT and then feeding them into a linear layer that classifies the outputs.
    3. A stacked Bidirectional LSTM model architecture by taking all the output hidden layers from BERT and then feeding them into the stacked bidirectional LSTM network, which pools all the representations of the [CLS] token embeddings from the last hidden state output, uses the output of the last LSTM cell and feeds it into a linear layer that classifies the outputs.
- Evaluating the final performance of the proposed models based on accuracy, precision, recall, F1-score, Matthews Correlation Coefficient (MCC) and AUC-ROC curve metrics on two publicly available test sets. Moreover, the training and validation loss as well as the accuracy are presented in plots to evaluate model performance according to the training and validation sets.
- Comparing the final performance of the proposed models to the performance of two competition winner models on the same test sets from previous related studies.

- Concluding with the findings and limitations of the paper as well as setting new aims for future work.

1.2 Thesis Structure

The research in this thesis is described according to the following chapters: Chapter 1 constitutes the introduction to the main topic and emphasizes on the reasons for conducting this type of research. Chapter 2 mentions related work done by different researchers and the various definitions of offensive language concerning the task. Chapter 3 focuses on a detailed technical explanation of BERT and LSTM networks as well as the different proposed model architectures. Chapter 4 describes the materials and experimental evaluation, which is classified into 3 sub-tasks: the first includes the experimental datasets, the second involves the experimental setup, whilst the results of the experiments along with the comparison of the models are illustrated in the third sub-task. Last but not least, chapter 5 concludes the work and presents its limitations, while chapter 6 suggests future methods.

2. RELATED WORK

In the earliest research conducted by Spertus (1997), a prototype system named *Smokey* was developed to automatically detect private abusive messages or flames sent through feedback forms on World-Wide Web pages. It combined Natural Language Processing and sociolinguistic observations to identify online comments that not only include insulting vocabulary, but also use it in an insulting manner. On account of the significant rise of user-generated content on social media, many studies have focused on identifying insulting messages on different social media platforms targeting either individuals or groups by combining Natural Language Processing (NLP) with machine learning. Even though most of them had begun investigating the English language, as it is rich in resources, the need to examine this problem in a multi-lingual context has emerged in the recent years. The surge in interest in exploring the present classification task has been demonstrated through a great number of conducted and published studies, which have utilized a variety of relevant terminologies ranging from *abuse*, *aggression*, *cyberbullying*, *hate speech* to *toxic* or *offensive language*. In the following paragraphs, several related studies using text classification with different terms are briefly described.

SemEval (*Semantic Evaluation*) is a series of international NLP research workshops whose purpose is to improve the current state of the art in semantic analysis as well as create high-quality annotated datasets in a variety of increasingly challenging problems in natural language semantics. It has been held since 1998 and continues until today. Each year's workshop gives prominence to a collection of shared tasks ranging from Named Entity Recognition, Classification to Sentiment Analysis. Participants are invited to develop and evaluate their approaches based on data provided by the organizers. All the submitted approaches are then presented and compared. The detection of offensive language in social media was initially introduced as a topic in the 6th shared task of SemEval carried out in 2019 under the name *OffensEval: Identifying and Categorizing Offensive Language in Social Media*. The task included three sub-tasks whose goals were to implement binary or multi-class text classification. The aim of sub-task A was to discriminate between offensive and non-offensive English tweets, while the aim of sub-task B was to identify the type of offensive tweets, whether they were targeted or not. The goal of sub-task C was to identify target of the offensive posts. A dataset containing 13.240 English tweets and a test set of 860 tweets, called *Offensive Language Identification Dataset* (OLID), were annotated according to the three sub-tasks and provided to the participants. The developed models varied from traditional machine learning, like SVM and Logistic Regression, to deep learning, such as CNN, RNN, BiLSTM, BERT. However, it was concluded that the best approaches utilized ensembles and deep learning models, namely BERT. (Zampieri et al., 2019). The 12th task of SemEval 2020 (OffensEval 2020) named as *Multilingual Offensive Language Identification in Social Media* was an extension of the previous task for other languages, such as Arabic, Danish, Greek and Turkish, to be studied as well. Each language was given its own dataset. A number of approaches across various languages and datasets was compared. It was revealed that the best performing machine learning methods employed different versions of BERT, like BERT, RoBERTa and mBERT (Zampieri et al., 2020).

Offensive language identification was also carried out in GermEval 2018 as well as in the 2nd task of GermEval 2019 for German tweets. An annotated dataset including about 8.500 tweets was provided. The task comprised of two sub-tasks requiring for binary and multi-class classification. The first sub-task aimed to differentiate between offensive and non-offensive tweets, whereas the second aimed to categorize the offensive tweets into profanity, insult and abuse. In GermEval 2018, the best systems employed traditional supervised machine learning methods, namely SVMs (Wiegand et al., 2018), while the best performance was achieved by versions of BERT in GermEval 2019 (Struß et al., 2019).

Based on the two previous GermEval shared tasks, the GermEval 2021 shared task focused on the identification of toxic, engaging and fact-claiming comments. The provided dataset comprised 4.188 Facebook posts derived from a German political talk show that was broadcasted in the national public television. It consisted of three binary classification sub-tasks, namely *Toxic Comment*, *Engaging Comment* and *Fact-Claiming Comment Classification*. The participants utilized the contextual embeddings from various pre-trained Transformer models and traditional machine learning algorithms (Risch et al., 2021). Moreover, Kaggle organized the Toxic Comment Classification Challenge[1] in 2018, which was an open competition that provided participants with comments from the English Wikipedia to be classified into six distinct categories, namely *toxic*, *severe toxic*, *obscene*, *threat*, *insult* and *identity hate*.

One of the most studied detection tasks concerned hate speech. For instance, the 5th task of SemEval 2019 addressed the problem of hate speech towards immigrants and women in Twitter in English and Spanish. Its two sub-tasks asked for binary classification, whether a post was hateful or not, and whether the target was generic or an individual (Basile et al., 2019). Moreover, Davidson et al. (2017) introduced the hate speech detection dataset containing over 24.000 English tweets labeled as non-offensive, hate speech and profanity. Apart from this, the unified evaluation framework consisting of seven heterogeneous Twitter classification tasks proposed as *TWEETEVAL* included hate speech detection and offensive language detection as two distinct tasks (Barbieri et al., 2020). According to Davidson et al. (2017), the need to differentiate the terms *hate* and *offensive* and divide them into sub-categories in order to assist the identification of insulting language has been highlighted by previous studies, since researchers tend to conflate one with another. The difficulty to distinguish between them is also underlined.

As far as the task of aggression detection is concerned, the aim of the *Shared Task on Aggression Identification* conducted as part of the *First Workshop on Trolling, Aggression and Cyberbullying (TRAC - 1)* at COLING 2018 was to develop a multi-class classifier that could differentiate between *Overtly Aggressive*, *Covertly Aggressive*, and *Non-aggressive* texts. The participants were provided with a dataset containing 15.000 aggression-annotated Facebook posts and comments in Hindi and in English for training and validation, while two different test sets were provided, one from Facebook and one from Twitter (Kumar et al., 2018). The best performing systems utilized deep learning methods, like CNN and LSTM, either by training on augmented data and pseudo labeled examples (Aroyehun & Gelbukh,

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

2018) or by using FastText word embeddings (Modha et al., 2018). Last but not least, Reynolds et al. (2011) as well as more recent studies (Islam et al., 2020) have addressed the issue of cyberbullying detection, while other work has focused on abusive language detection (Janardhana et al., 2020; Akhter et al., 2021). These studies also employed both machine learning, like SVM, Naïve Bayes, Logistic Regression, and deep learning methods, like CNN and LSTM, to address the problem in social media.

Although each of the aforementioned studies have focused on a specific type of abuse or offense, one can observe that there are many features in common as far as the purpose, the type of content and the targets are concerned. In the present thesis, the term offensive language is used in accordance with the two SemEval Offensive Language Identification Tasks. It can be considered as a broad umbrella term comprising of various types of user-created content adopted by the NLP community aiming to harass, insult, belittle and threat individuals or groups. The paper utilizes the OLID datasets, whose hierarchical annotation of labels aims to capture all these forms of insulting expression (Zampieri et al. 2019). While these studies have approached this problem from different angles, it has been revealed that deep learning models as well as ensemble models manage to achieve the best performance and offer state of the art results. By taking this into account, the proposed model methods utilize the contextual word embeddings from $BERT_{LARGE}$ so as to leverage the contextual properties of each tweet by fine-tuning its pre-trained embeddings on the training data. In this way, both the existing knowledge of the pre-trained embeddings and the patterns of the English tweets can be exploited.

3. MACHINE LEARNING METHODS

In this thesis, the problem of offensive language in the Twitter social media platform is addressed as a supervised learning, binary classification task with the utilization of machine learning methods. More particularly, deep learning methods are employed, such as BERT and LSTM networks, with a view to achieving highly accurate results. In this chapter, the proposed model architectures are explained in detail.

3.1 BERT

BERT stands for Bidirectional Encoder Representations from Transformer and it was introduced by researchers at Google AI Language (Devlin et al.) in late 2018. It is a language model that learns simultaneous contextual information from a sequence of words from both directions. It has made quite a splash in the machine learning community, as it has managed to achieve state-of-the-art results on 11 NLP tasks, like Question Answering and Natural Language Inference.

It provided a major breakthrough due to its innovative methods and architecture that allow for multi-layer bidirectional Transformer training. More particularly, BERT utilizes the bidirectional nature of the encoder stacks, since it consists of several Transformer encoders stacked together. Each Transformer encoder encapsulates two sub-layers, a self-attention layer and a feed-forward layer. It takes as input a token sequence, which may constitute a single sentence or a pair of sentences. BERT was pre-trained on unlabeled data extracted from BooksCorpus (800 million words) and Wikipedia (2.500 million words) by utilizing two unsupervised tasks, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Masked Language Modeling (MLM) masks 15% of the words in an input sequence and, then, endeavors to predict the masked words by taking into consideration the context provided by both left and right surrounding non-masked words at the same time. Out of the 15% of the selected to be masked tokens, 80% of the tokens are replaced with the token [MASK], while 10% of the time the tokens are replaced with a random token or remain unaffected. Furthermore, Next Sentence Prediction (NSP) tries to understand the relationship between a pair of sentences. That is to say, it aims to find out whether the second sentence is the subsequent sentence in a pair of sentences or whether it is just a random sentence from the corpus. During training, there is a 50% chance of both cases. NSP is mainly employed for relevant tasks such as Question Answering (Khalid, 2019).

BERT requires a specific input representation in order to understand the context and make predictions. The input representation needs to be the sum of the token, the segment and the positional embeddings (Devlin et al., 2019). First, the token embeddings comprise two special tokens, the [CLS] and the [SEP] token. The [CLS] token stands for classification token and is added at the beginning of every sequence. This token is considered significant for classification tasks. The [SEP] token informs BERT which token belongs to the first and which token to the second sentence. This token is important for a Next Sentence Prediction or a

Question Answering task. In case there is only one sequence, this type of token is appended to the end of the sequence. Secondly, the segment embeddings are markers that are added to each token and indicate which tokens are included in sentence A and which tokens in sentence B. In other words, they assist BERT in distinguishing between sentences. Thirdly, the positional embeddings are also added to each token to demonstrate its position in the sentence. The figure below presents the necessary input embeddings for BERT according to its developers (Devlin et al., 2019).

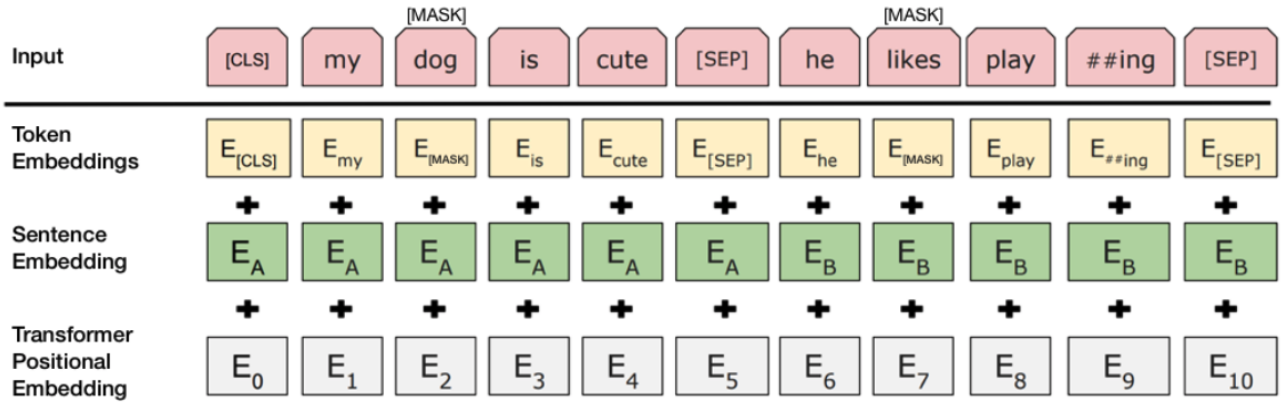


Figure 1. BERT input representation from Devlin et al. (2019)

Two model sizes were released, namely $BERT_{BASE}$ and $BERT_{LARGE}$. $BERT_{BASE}$ has 110 million parameters in total, since it consists of 12 layers with 768 hidden size and 12 self-attention heads. On the other hand, $BERT_{LARGE}$ has 340 million parameters in total, as it comprises 24 layers with 1024 hidden size and 16 self-attention heads. According to the authors (Devlin et al., 2019), it was demonstrated that increasing the model size has a positive effect on the model performance due to the fact that $BERT_{LARGE}$ outperformed $BERT_{BASE}$ across the 11 NLP tasks. Additionally, BERT comes in two versions, Cased and Uncased. In BERT Uncased, the text is lowercased before tokenized with the WordPiece tokenizer, whereas in BERT cased, the text is maintained with its accent markers as it is.

BERT has been provided for free to the NLP community in both Pytorch and TensorFlow libraries on Hugging Face so that it can be utilized for fine-tuning. During fine-tuning, BERT is initialized with its pre-trained parameters, which are fine-tuned using labeled data from downstream tasks, for instance Named Entity Recognition, Question Answering and classification. By adding a specific task component to the end of BERT and fine-tuning the result, the method of transfer learning is achieved. Transfer learning with pre-trained models like BERT is often opted in favor of building a deep learning model from scratch, since the pre-trained parameters encode enough information to offer faster model development with minimal task-specific adjustments and less amount of data, resulting in achieving state-of-the-art results for a wide variety of tasks. Many Transformer models were developed based on BERT due its successful performance, for instance RoBERTa and ALBERT.

3.2 LSTM Networks

LSTMs stands for Long Short-Term Memory networks and are a kind of Recurrent Neural Networks (RNNs). They were first introduced by Hochreiter & Schmidhuber in 1997. They are considered a popular deep learning method in NLP, since they are designed to capture long-term dependencies unlike standard RNNs. In other words, they have the ability to remember previous information and utilize it to process a current input. They have achieved outstanding performance on a variety of problems, such as machine translation and speech recognition.

Although the LSTMs have the chain like structure resembling the RNNs, the repeating module includes four neural network layers interacting in a very special way instead of one. The LSTM networks possess a long-term memory cell known as cell state through which information flows unchanged. The cell state is represented by $C_{(t-1)}$ and $C_{(t)}$ for the previous and current timestamp respectively. Apart from the cell state, LSTMs have a hidden state, the short-term memory, where $H_{(t-1)}$ represents the hidden state of the previous timestamp and H_t the hidden state of the current timestamp. The cell state is regulated by three gates, the forget, the input and the output gate. These gates optionally let the information flow in and out of the cell. The forget gate, which is the first gate, determines whether the information coming from the previous state is to be remembered or is irrelevant and can be forgotten. A sigmoid function is applied over the gate that outputs a number between 0 and 1 denoting whether the information will be removed or maintained respectively. The forget gate is then multiplied with the cell state of the previous timestamp in order to keep or not information. Next, the input gate determines what type of new information to store to the cell state from the current input. Its sigmoid function decides the values that will be updated and its tanh layer creates a vector for new candidates that will be added to the current cell state. After that, the output gate uses its sigmoid function to determine what to output from the cell state. The tahn activation function is used to convert the values of the input between -1 and 1. Finally, the result is multiplied with the outcome of the output gate (Saxena, 2021).

Apart from the original LSTM model which consists of a single hidden LSTM layer followed by a standard feedforward output layer, an extension has developed, the stacked LSTM, which includes multiple hidden LSTM layers with each layer containing multiple memory cells. Thus, a much more deeper model architecture is created with the potential to solve challenging sequence prediction problems. In addition, Bidirectional LSTM networks (BiLSTM) are an upgraded version of LSTM networks. Unlike the standard model architecture, BiLSTM adds one more LSTM layer that reverses the direction of information flow to move backward. This means that the model is capable of utilizing information by processing data in both directions. The outputs from both these hidden layers, the one with processing information forward and the other backward, can be combined in several ways, like average, sum, multiplication, or concatenation. In this way, BiLSTM networks are able to produce more meaningful outputs, since they have gained complete information about every point in a sequence, everything before and after it by combining LSTM layers from both directions. Nevertheless, BiLSTMs require much more time for training than the standard LSTMs (Zvornicanin, 2022).

3.3 Proposed Model Architectures

BERT has been widely used for fine-tuning downstream tasks by adding an additional output layer, which takes representations only from the last layer of the pre-trained language model as the default input. Nevertheless, this fine-tuning approach completely disregards significant semantic and syntactic knowledge, which is contained in the other layers. Considering the multi-layer structure of BERT, different layers are able to capture various levels of representations and, hence, encode very different kinds of linguistic information, such as surface features in the lower, syntactic features in the middle and semantic features in the higher layers. The authors of BERT (Devlin et al., 2019), followed a feature-based approach by extracting the contextual embeddings from different layers of $BERT_{Base}$ and providing them as input to a BiLSTM for a Named Entity Recognition task. From their experiments, it became evident that the concatenation of the last four hidden layers was the best pooling strategy, since it produced the highest F1 score. The different pooling strategies the authors used along with the number of layers and F1 scores on the development set are presented in figure 2 below (Alammar, 2018).

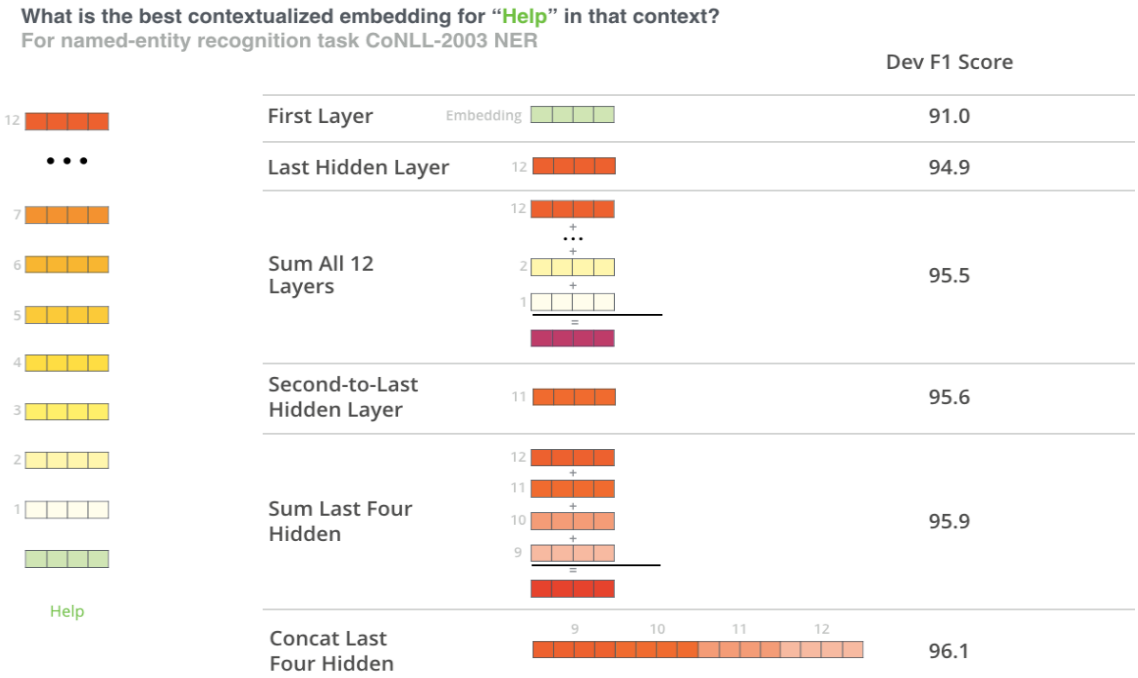


Figure 2. Feature-based approach with BERT for CoNLL-2003 NER from Alammar (2018)

Consequently, inspired by the feature-based approach of the original BERT paper (Devlin et al., 2019), the three proposed model architectures of the thesis show the potential of utilizing information from different BERT layers in order to facilitate the fine-tuning process and achieve state-of-the-art results.

The first proposed model consists a BERT basic custom classifier structure (***BERT Classifier Last Hidden***). From the embeddings of each sequence contained in the last hidden layer (last hidden state) of $BERT_{LARGE}$ with output shape [batch size, max seq len, hidden size], it

takes the first position token embeddings that capture the entire context and are meant for classification, the [CLS] embeddings ([batch size, hidden size]). Then, a dropout layer to prevent overfitting is applied to those embeddings. Finally, a linear layer with dimensions of the hidden size of $BERT_{LARGE}$ (1024) and the number of classes (1) is applied to the output from the dropout layer, which is responsible for classifying the tweets into offensive and not offensive. The final output has dimensions [batch size, output label number]. The proposed architecture is illustrated in figure 3 along with the output dimensions of each layer.

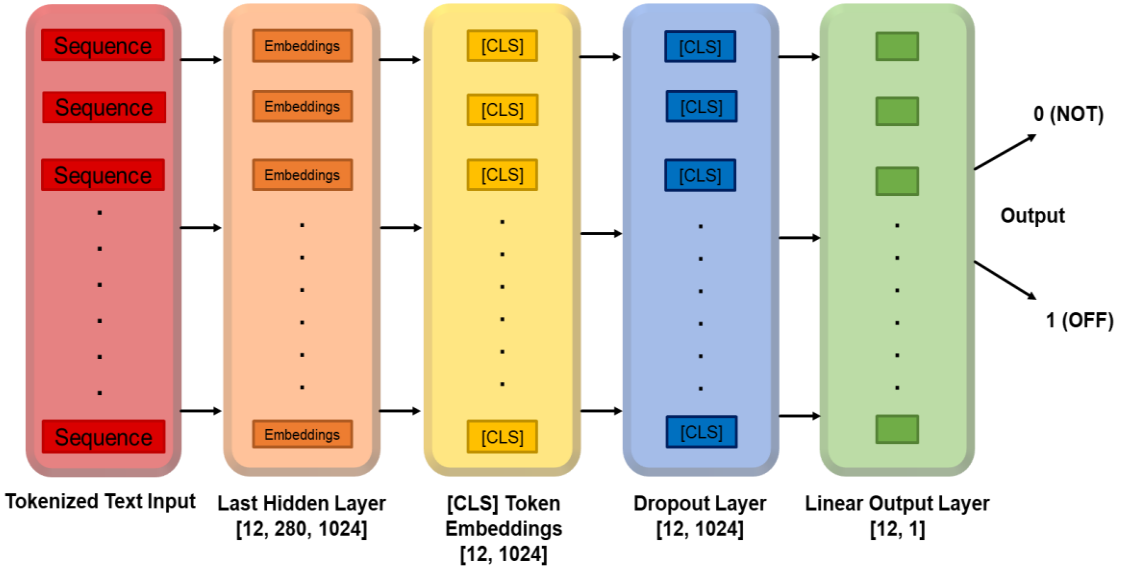


Figure 3. Architecture diagram of *BERT Classifier Last Hidden*

The second proposed model consists another BERT custom classifier structure (***BERT Classifier Concat Last 4 Hidden***) that takes the output from all the hidden layers of $BERT_{LARGE}$ ([initial embeddings + 24 BERT layers, batch size, max seq len, hidden size]) and concatenates only the last four layers into one with output dimensions [batch size, max seq len, hidden size * 4]. The [CLS] token embeddings are taken from the last four hidden layers with output dimension [batch size, hidden size * 4] and a dropout layer is applied. Finally, a linear layer with dimensions of four times the hidden size of $BERT_{LARGE}$ (1024 * 4) and the number of classes (1) is applied to the output from the dropout layer, which implements the binary classification. The final output has dimensions [batch size, output label number]. This was one of the pooling strategies that performed best according to the authors of BERT. To enable getting all the hidden states from the Transformer model, the parameter *output_hidden_states = True,* when defining the model from the Transformers class, was implemented. The proposed architecture is illustrated in figure 4 along with the output dimensions of each layer.

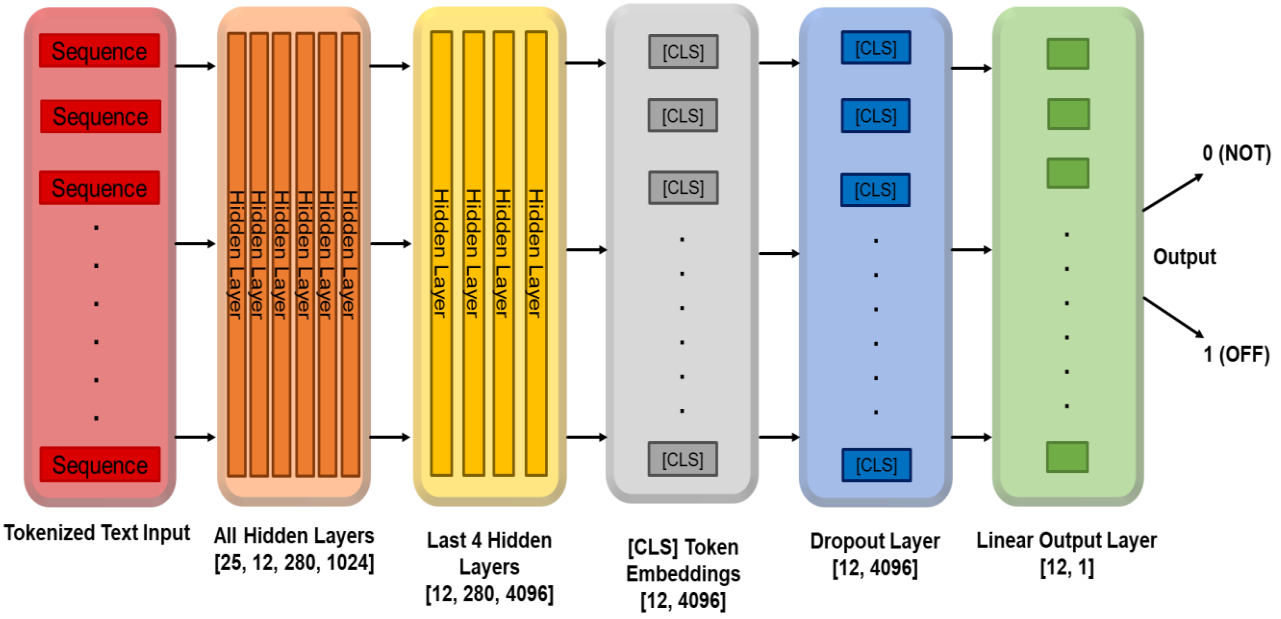


Figure 4. Architecture diagram of *BERT Classifier Concat Last 4 Hidden*

The third proposed model is a Bidirectional Stacked LSTM network (***BERT Bi-Stacked-LSTM***) utilizing LSTM pooling. It is based on the model architecture introduced by Song et al. (2020) for aspect-based sentiment analysis, an extension of it is developed though, as the proposed model is both stacked and bidirectional. Due to the fact that the LSTM network is developed as bidirectional with 2 stacked LSTM hidden layers, it is able to process the input multiple times and retain information not only sequentially, but also from both directions. Firstly, it takes the hidden states of the [CLS] token from all layers of $BERT_{LARGE}$ ([initial embeddings + 24 BERT layers, batch size, max seq len, hidden size]). The dimensions of the hidden states are squeezed and converted into [batch size, number of layers, hidden size] to fit into the Bi-Stacked-LSTM layer. After that, the LSTM is used to connect the [CLS] token representations resulting in getting an output of the last LSTM cell as the final representation with output dimensions [batch size, 24 $BERT_{LARGE}$ layers, max seq len * 2]. A dropout layer is applied to the LSTM output. Finally, a linear layer with dimensions of two times the maximum sequence length (280 * 2) and the number of classes (1) is applied to the output from the dropout layer, which predicts the offensive and not offensive tweets. The final output has dimensions [batch size, output label number]. Once again, the parameter *output_hidden_states* was set to *True,* when defining the model from the Transformers class. The proposed architecture is illustrated in figure 5 along with the output dimensions of each layer.

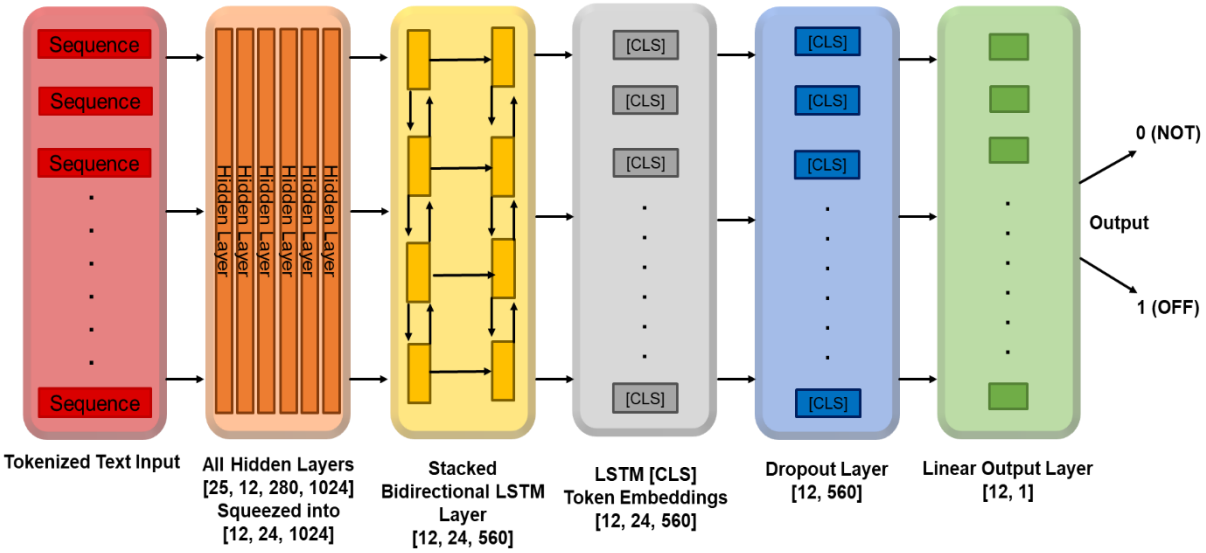


Figure 5. Architecture diagram of *BERT Bi-Stacked-LSTM*

4. MATERIALS & EXPERIMENTAL EVALUATION

In this chapter, the materials used in this study are presented, including data acquisition and preparation, the model parameters, the evaluation metrics as well as the results from the experiments and comparison with previous related work.

4.1. Experimental Datasets

The data acquisition was followed by a number of data wrangling and preprocessing techniques in order to create cleaned, organized and appropriate in format datasets that were used either for training or for testing. Four different training and two test datasets were collected in total, containing English texts from Twitter. The *Pandas* library was utilized for this part, which is considered the most popular library for data manipulation and analysis.

4.1.1. Data wrangling

The data wrangling techniques ranged from dropping duplicates and index, replacing label values to merging Twitter data from different sources into one dataset. More particularly, data exploration was the first step applied to get the basic information about each dataset, for instance the number of columns, their values and the size (shape) of the dataset. Then, certain columns were renamed, changed order of appearance, while other columns were completely removed, as they were considered inessential. After that, it was time to check for empty or duplicate values and remove (drop) them. The duplicates were dropped except from the first occurrence. The index was removed as well. Finally, the values included in the columns were renamed wherever needed. Each dataset was processed with the appropriate steps so that it would take the form of a dataframe comprising two columns named *tweet* and *label*. The first column included the texts and the second column their corresponding categorical labels, *OFF* (*Offensive*) and *NOT* (*Not Offensive*). In the following paragraphs, the general information regarding the datasets used for model training and evaluation are presented.

4.1.2. Datasets

The first dataset named *Offensive Language Identification Dataset (OLID)* was introduced by Zampieri et al. (2019) and was originally used in task 6 included in the *SemEval* competition that took place in 2019, the *OffensEval: Identifying and Categorizing Offensive Language in Social Media*. This task was split into three individual sub-tasks: A) *Offensive Language Detection* B) *Categorization of Offensive Language* and C) *Offensive Language Target Identification*. The first sub-task aimed to discriminate between the offensive and not offensive Twitter posts with the labels *Offensive (OFF)* and *Not Offensive (NOT)*. Every instance in the OLID dataset was firstly assigned its own label according to this task. In case an instance belonged to the *OFF* category, it was also annotated with a different label in the next sub-task. The second sub-task's objective was to discriminate between instances containing an insult/threat targeted to an individual, group, or others with the label *Targeted*

Insult (TIN), and instances including non-targeted profanity and swearing with the label *Untargeted (UNT)*. The third sub-task focused only on the tweets in the *TIN* category and intended to differentiate the targets that received offenses into *Individual (IND)*, *Group (GRP)* and *Other (OTH)*. For the binary classification task in my thesis, only the column of the first sub-task was maintained, whereas the other two columns were discarded as unnecessary. The dataset consisted of 13.240 annotated tweets, out of which 33 duplicates were removed. Consequently, among the 13.207 unique tweets, the 4.392 were labelled as *OFF* and the 8.815 as *NOT*.

The second dataset was found in the Github repository for the *TweetEval*[2] (Barbieri et al., 2020), which comprised of seven heterogenous tasks in Twitter unified into the same benchmark. From the dataset for the Offensive Language Identification task, which came into training, validation and test splits, only the training and validation texts along with their labels were used. They were concatenated resulting in creating a new dataset. This dataset consisted of 13.240 tweets, out of which 38 duplicates were removed. Therefore, among the 13.202 unique tweets, the 4.394 were labelled as *OFF* and the 8.808 as *NOT*.

The third dataset was found available on the Kaggle platform[3] and it was originally used in research by Davidson et al. (2017) concerning hate speech and offensive language detection. The dataset consists of 24.783 tweets that can be considered racist, sexist, homophobic, or generally offensive. It is classified in three labels, namely *hate speech (0)*, *offensive language (1)* and *neither (2)*. As I utilize the term offensive language as an umbrella term including all types of offensive and hateful texts, the instances classified in the 0 and 1 classes were renamed as *OFF*, while those classified in class 2 were renamed as *NOT* in order to take the same form of the two previous datasets. The rest of the columns were removed, as they were deemed unnecessary. The order of the columns was altered as well. There were no duplicates to remove, thus the dataset contained 20.620 *OFF* and 4.163 *NOT* values.

The fourth dataset was also found available on the Kaggle platform[4] and contains 56.745 tweets regarding hate speech and offensive language that are classified into two labels, *toxic (1)* or *not (0)*. The id column was removed and the columns changed order. The instances classified in category 1 were renamed as *OFF* and in 0 renamed as *NOT*. From the total number of tweets, 2.432 duplicates were removed. Hence, among the 54.313 unique values, 23.924 were classified in the *OFF* and 30.389 in the *NOT* labels.

Taking into consideration the fact that previous studies had experimented with a relatively small amount of labelled data for this task, the decision to search for more data in order to increase as much as possible the training dataset was taken, so that the models could offer even improved performance. Therefore, the four aforementioned datasets were concatenated into a single dataframe containing 105.505 tweets. Among these tweets, 24.783 duplicates were removed, since a lot of duplicate offensive data were found. As a result, the combined training dataset consists of 80.722 unique tweets, out of which the 38.92% of texts (31.419) were labelled as *OFF*, whereas the 61.08% of texts (49.303) were

[2] https://github.com/cardiffnlp/tweeteval
[3] https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset/discussion/235251
[4] https://www.kaggle.com/datasets/ashwiniyer176/toxic-tweets-dataset

labelled as *NOT*.

At this point, it is clearly demonstrated that the newly combined training dataset is imbalanced, since the not offensive tweets represent the majority class. The deficiency of offensive data has always been the case in most datasets so far and one of the main complications concerning the present binary classification task. Models tend to be biased towards the majority class and, hence, are not able to predict many instances belonging to the other class. The need to train models with enough offensive data in order to detect the offensive language has never been more critical. For this reason, a resampling technique was decided to be applied in order to handle the imbalance in the dataset. The *resample* function was used from *Sklearn* library. A balancing method called *down-sampling* was employed. That is to say, the majority class (*NOT*) was reduced to the exact number of minority class (*OFF*) values with a view to achieving dataset balance. As a consequence, the balanced dataset consists of 62.837 in total, 31.419 tweets in each class. From the experiments, it will become evident whether the performance of the models will benefit from the action to balance the training dataset. Last but not least, the categorical values *OFF* and *NOT* were mapped with the numerical labels 1 and 0 respectively to facilitate the training process.

The test datasets were also named *Offensive Language Identification Dataset (OLID)* and they were originally used in the *SemEval* competitions that took place in 2019 and 2020 (Zampieri et al., 2019). The test OLID 2019 was used for testing in task 6 named *OffensEval: Identifying and Categorizing Offensive Language in Social Media*. It contained 860 tweets, out of which 240 were labelled as *OFF* and 620 as *NOT*. The test OLID 2020 was much increased in size than the previous one and was used for testing in task 12 named *Multilingual Offensive Language Identification in Social Media*. It comprised 3.887 tweets in total, 1.080 *offensive (OFF)* and 2.807 *not offensive (NOT)*. As *OFF* were labelled the tweets involving profanity or a targeted offense that may be direct or indirect, while as *NOT* the tweets that do not belong in the other class.

Table 1: Class distribution of each dataset

DATASETS	OFFENSIVE VALUES (OFF)	NOT OFFENSIVE VALUES (NOT)
OLID training 2019	33.26% (4.392)	66.74% (8.815)
TweetEval 2020	33.28% (4.394)	66.72% (8.808)
Kaggle Hate-Offensive-Neither	83.20% (20.620)	16.80% (4.163)
Kaggle Toxic or Not	44.05% (23.924)	55.95% (30.389)
Combined training dataset	38.92% (31.419)	61.08% (49.303)
Combined Balanced training dataset	50.00% (31.419)	50.00% (31.419)
OLID test 2019	27.91% (240)	72.09% (620)
OLID test 2020	27.78% (1.080)	72.22% (2.807)

4.1.3. Data Pre-processing

The data pre-processing techniques ranged from omitting usernames and links to tokenizing and lowercasing the words. More specifically, the *BeautifulSoup*, the latest version of *Emoji*[5] (Kim, et al, 2022) and the *Ekphrasis*[6] (Baziotis, et al., 2017) libraries as well as many regular expressions were utilized for this part. They were implemented through a function encompassing all the preprocessing steps and were applied in the *tweet* column of the training and test dataframes. Unlike data preprocessing methods in previous related work, the punctuation and the emojis were maintained as they express sentiment and sarcasm and contribute to the contextual information of the tweets that is provided to the models.

The *Emoji* library contains a list of numerous Unicode emoji characters and sequences, with images from different vendors, like Facebook and Twitter, in several languages, such as English, French and German. The *Ekphrasis* library was developed by Baziotis, et al. (2017) as a part of the text pre-processing pipeline for their submission in task 4 concerning *Sentiment Analysis in Twitter (English)* of the *SemEval* competition, which was conducted in 2017. It uses word statistics based on English Wikipedia and 330 million tweets. It comes with the *TextPreProcessor* function including tools that can tokenize, normalize, annotate and segment words, correct the spelling of words and expand the contracted words.

Table 2: Two examples of tweets included in each dataset

DATASETS	INSTANCES OF OFFENSIVE TWEETS
OLID training 2019	*Canada doesn't need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo*
	if you aint bout that Murder Game pussy nigga shut up
TweetEval 2020	*SHE IS A FUCKING MESS!! I HATE HER SO MUCH.*
	you couldn't catch me letting some saggy whitey bitch talkin like this to me 😭 bitch eat dirt
Kaggle Hate-Offensive-Neither	*fuck no that bitch dont even suck dick " 😂😂😂 the Kermit videos bout to fuck IG up*
	we dont trust these niggas all these bitches
Kaggle Toxic or Not	*shot inside a pussy office woman*
	lying @user and his disingenuous excuses for #deviant remarks about michelle & #barackobama:
OLID Test 2019	*#ArianaAsesina? Is that serious?! Holy shit, please your fucking assholes, don't blame someone for the death of other one. She is sad enough for today, don't you see? It isn't fault of none, he had an overdose and died. End. Stop wanting someone to blame, fuckers.*
	@USER How is she hiding her ugly personality. She is the worst.
OLID Test 2020	*@USER Yeah - respect for the country. No one respects you though you fat traitor*
	@USER Listen bitch I'm still jealous of @USER for being able to suck ur cock whenever lol

[5] https://pypi.org/project/emoji/
[6] https://github.com/cbaziotis/ekphrasis

4.1.4. Offensive Words in Offensive Tweets

In a study conducted by Cachola et al. (2018) pertaining to sentiment analysis of vulgar tweets, a vulgarity lexicon[7] was employed in order to identify the vulgar tweets in their dataset. This lexicon consists of 349 entries involving general offensive words, slurs and sex-related terms, out of which the 82 were manually removed as they were considered to be unambiguously vulgar. Moreover, the researchers created a list of regular expressions in a txt file so as to identify several common intentional spelling variations in the vulgar terms, for instance elongated words, plurals, adjectives ending in *-ed* or *-ing*. They counted the number of vulgar words and the vulgar word frequency, while viewing the most common vulgar word occurrences in their dataset. They recommended and compared three different ways to account for vulgarity by taking into consideration certain vulgarity features. Thus, they introduced masking, token insertion and concatenation during their data preprocessing by creating additional columns in the dataset representing each vulgarity feature. More specifically, the masking method applied the removal of a vulgar word and its replacement with the special token <*VG*> to denote the occurrence of that word at the particular sentence position, for example This is the <*VG*>. By applying this method, the model assumed that all vulgar words constituted the same token, the lexical variances and the different vulgar expressions along with their meanings were removed though. In the token insertion method, the special token <*VG*> was added to the right of an identified vulgar word, such as This is the shit <*VG*>, thus retaining each word's meaning. However, it provided the same representation as other relevant words. In the concatenation method, the number of vulgar words in each sentence was counted and added in an additional dataframe column to be used as a feature in concatenation with the tweet representation from the hidden states of their proposed Bi-LSTM by creating the input *[hidden state, number of vulgar words]*. Nevertheless, in this way, the contextual information concerning the sentence position of a vulgar word was lost. Four models were trained, each model containing one vulgarity feature and a baseline model without. The results demonstrated that the Bi-LSTM model with the token insertion as well as the bi-LSTM model with the concatenation method increased model performance, since they produced the lowest Mean Absolute Error (MAE).

By taking into account the improved model performance, I decided to follow the same procedure of acquiring just one of the vulgarity features, the token insertion, for my training dataset aiming to increase the probability of the models predicting more offensive tweets. Therefore, I employed the vulgarity lexicon and the regular expressions from the aforementioned study with a view to looking for and detecting the offensive words only in the tweets that were labelled as *OFF*. The special token <*OFF*> was added right next to an offensive word, for instance he is a dumbass <*OFF*>. It is noteworthy to mention that approximately 62.2% of the 31.419 offensive tweets contained terms from the vulgarity lexicon (19.546 tweets). The ten most frequent words along with their number of occurrences involved *bitch* (8.505), *bitches* (3.146), *pussy* (2.196), *shit* (1.953), *hoe* (1.907), *ass* (1.903), *fuck* (1.824), *nigga* (1.291), *fucking* (905) and *faggot* (435). This procedure was encompassed in the pre-processing stage and was applied after the rest pre-processing

[7] GitHub - ericholgate/vulgartwitter: Corpora for vulgar and censored tweets annotated for sentiment

methods, but not to the test sets.

4.1.5. Data Preprocessing Methods: Overview

The preprocessing methods were applied sequentially as follows:

1. The UTF-8 Byte Order Mark (BOM) was removed. It is a sequence of bytes (EF BB BF) that allows the reader to identify a file as being encoded in UTF-8.
2. The *BeautifulSoup* library was used to decode data like certain emoticons so that they could be converted by the *Emoji* library.
3. The retweet (*RT*) was replaced with whitespace.
4. The usernames and the URL links as well as the words *url*, *html* and *http* were replaced with whitespace.
5. The emojis were converted from pictographs to their textual representation, for example the 😀 emoticon was turned into *grinning face*.
6. The *&* and *&* were replaced with the word *and* as well as the ASCII encoding *’* apostrophe with the *‘* apostrophe in encoding UTF-8 so that the *TextPreProcessor* can identify and expand the contracted words.
7. The consecutive non-ASCII characters were replaced with whitespace.
8. All the extra whitespace created with the replacements was removed.
9. The hashtags were segmented by separating the # from the word, for example the hashtag *#Liberals* was turned into *liberals*, and the special tokens *<hashtag>* and *</hashtag>* were added on both sides of a hashtag.
10. Correction spelling and correction of elongated words was applied.
11. The contracted words were unpacked, for example *can’t* was turned into *can not*.
12. The *SocialTokenizer* was used to tokenize and lowercase all words in the tweets.
13. Only the offensive words in the offensive tweets were counted based on the vulgarity lexicon and the special token *<OFF>* was added to the right of each word. This step was not applied to the test data.

Table 3: Several examples of tweets before and after pre-processing

ORIGINAL TWEET	PRE-PROCESSED TWEET
!!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out…	*! ! ! ! as a woman you should not complain about cleaning up your house . and as a man you should always take the trash out . . .*
*Canada doesn’t need another CUCK! We already have enough #LooneyLeft #Liberals f**king up our great country! #Qproofs #TrudeauMustGo*	*canada does not need another cuck ! we already have enough <hashtag> looney left </hashtag> <hashtag> liberals </hashtag> f**king up our great country ! <hashtag> qproofs </hashtag> <hashtag> trudeau must go </hashtag>*
fuck no that bitch dont even suck dick " 😂😂😂 the Kermit videos bout to fuck IG up	*fuck no that bitch dont even suck dick " face with tears of joyface with tears of joyface with tears of joy the kermit videos bout to fuck ig up*
@JetsAndASwisher @Gook____ bitch fuck u http://t.co/pXmGA68NC1" maybe you'll get better. Just http://t.co/TPreVwfq0S	*: bitch fuck u maybe you will get better . Just*

4.2. Experimental Setup & Evaluation

The proposed machine learning methods of the thesis were implemented in Python programming language. The experiments were conducted on a Google Colaboratory (Colab) Pro notebook by utilizing GPU Tesla P100-PCIE-16GB. The experiments for the offensive language binary text classification task were carried out using the Pytorch library. To exploit the benefits of transfer learning by fine-tuning $\text{BERT}_{\text{LARGE-Uncased}}$, the *BertConfig*, *BertModel* and *BertTokenizer* were utilized from the Transformers class provided by Hugging Face[8].

To begin with, several requirements had to be met in order to start experimenting. The data, which were uploaded to my Google Drive, were imported into Google Colab by mounting the Drive. Packages of libraries like *transformers, torchtext, emoji, ekphrasis* and *unidecode* were essential to be installed. Apart from this, many classes and functions were imported from libraries including *pandas, sklearn, numpy, seaborn, matplotlib, glob, time, tqdm, random, nltk, torch, torchtext, transformers, ekphrasis, emoji and unidecode*.

4.2.1. Data Preparation

BERT, as all pre-trained models, requires a specific in format input for both training and evaluating. It takes a sequence that may consist of either one sentence or a pair of sentences. The special token [CLS] is always added at the start of each sequence as well as the [SEP] token which distinguishes between two sentences. Both special tokens are required in any case, since they are the expected input by BERT. The input sequences are always padded so that they have the same length. BERT can take up to 512 tokens in a sequence, but this maximum sequence length can be adjusted accordingly. In the event of a sequence including less tokens, the special token [PAD] with is corresponding id number 0 are used to cover the empty tokens until they reach the maximum sequence length. In case a sequence includes more tokens, then the sequence is truncated to fit the maximum sequence length.

BERT uses its own word-piece tokenizer, *BertTokenizer*, which is able to break words into sub-words, for instance the word *cleaning* would create the tokens *clean* and *##ing*. In this way, the model manages to recognize more familiar words contained in its vocabulary, while handling unknown words by encoding them with the [UNK] (unknown) token. In this thesis, since the special tokens <*OFF*>, <*hashtag*> and <*/hashtag*> were added in the sentences during pre-processing, they were added as tokens to the tokenizer's vocabulary as well, so that it won't split them into sub-words. Therefore, the token embeddings were resized resulting in $\text{BERT}_{\text{Large}}$ containing 30.525 tokens in its vocabulary and hidden size of 1024. Moreover, *BertTokenizer* provides the function called *encode_plus* that converts the input sequence into the necessary form understood by BERT. The result of this function is a dictionary containing the encoded tokenized sequence in input ids as well as other optional additional information, the token type ids (segment mask) and the attention mask. Firstly, the input ids are a sequence of integers that identify the index number of each input token included in the tokenizer's vocabulary. Moreover, the token type ids are only essential for tasks that involve sentence pairs like Question Answering, since they comprise a sequence

[8] https://huggingface.co/docs/transformers/model_doc/bert

of 0s and 1s to assist in differentiating between the first and the second sentence respectively. Thus, the token type ids are not necessarily needed for classification tasks. Finally, the attention mask, which is optional, consists of a sequence of 1s and 0s to differentiate between the actual tokens and the padding tokens respectively.

For the proposed classification task, the *encode_plus* function was called to create the required input for BERT and return only the input ids and the attention mask in Pytorch tensors. It added the special tokens [SEP] and [CLS] and padded the sequences according to the maximum sequence length. Even though the maximum sequence length of the training set was 415 tokens, it was set to 280 tokens to be in line with the established character limit in Twitter to enable more expression since November 2017. At first, experiments were conducted with 415 maximum sequence length, however, they offered the same results as 280 and twice the amount of training time. The function was included in a class inherited from the torch Dataset module (*torch.utils.data.Dataset)* that also returned the single value of 1 or 0 denoting the labels *OFF* and *NOT* respectively. An example of the keys included in the dictionary of the function is shown in table 4.

Table 4: Example of an encoded sequence from *encode_plus* function

ENCODED EXAMPLE OF A SEQUENCE
! ! ! ! as a woman you should not complain about cleaning up your house . and as a man you should always take the trash out . . .
Input ids: *tensor ([101, 999, 999, 999, 999, 2004, 1037, 2450, 2017, 2323, 2025, 17612, 2055, 9344, 2039, 2115, 2160, 1012, 1998, 2004,1037, 2158, 2017, 2323, 2467, 2202, 1996, 11669, 2041, 1012, 1012, 1012, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …]*
Attention mask: *tensor ([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …]*

The combined balanced data were first preprocessed using the steps analyzed in the first sub-task of Section IV. Then, the data were stratified split, so that 80% of instances were allocated for parameter fine-tuning and training (training set), while 20% of instances were used for evaluating the model performance after each epoch (validation/development set). The same preprocessing steps were applied to the two test sets as well, except from the insertion of token <*OFF*>. The two test sets were also concatenated into one combined test set to present the overall results and facilitate model performance comparison. Last but not least, the train, validation and test dataloaders were created using the *torch.utils.data.DataLoader*. The sentences contained in each DataLoader with batch size 12 are presented in table 5.

4.2.2. Hyperparameters of Models

Hyperparameter tuning is a significant stage when customizing pre-trained models for downstream tasks. For the proposed binary classification task, different parameters ranging from the number of epochs, early stopping patience, batch size, learning rate, dropout, to weight decay, were altered and adjusted during experimenting. By taking the trial-and-error approach, the optimal number of epochs in the experiments was set to 10. The early stopping patience, which prevents from overfitting, was configured and set to 3. This means that the training process is terminated in case the validation loss fails to improve for three consecutive evaluations. Although Devlin et al. (2018), recommend using a batch size of 16 or 32 for fine-tuning, the batch size was set to 12 since it was the largest batch size that the GPU memory could handle. The Binary Cross-Entropy Loss with Logits (*BCEWithLogitsLoss*) was utilized, as it includes a sigmoid layer and is a more numerically stable version compared to using a plain Sigmoid function followed by the Binary Cross-Entropy Loss (*BCELoss*). The *AdamW* was selected as the optimizer with the default betas 0.9 and 0.999, whereas the epsilon was set to 1e-2 (0.01) instead of the default 1e-8 because it led to very early overfitting no matter the other hyperparameters. The weight decay was also set to the default 1e-2 (0.01). The learning rate was set to 2e-5 (0.00002), which was one of the learning rates that were suggested by the developers of BERT (Devlin et al., 2018). According to Sun et al. (2019), a low learning rate such as 2e-5 is essential to make BERT deal with the catastrophic forgetting problem in transfer learning, which means that the pre-trained knowledge is deleted during learning of new knowledge. To avoid exploding gradients, the value of gradient clipping norm was set to 1.0. The dropout regularization to avoid overfitting issues was assigned as 0.1. The warm-up steps were set to 0, while the training steps, which are the number of times the batches pass to train the model, were calculated as 41.900. It was the result of the multiplication between the train dataloader and the number of epochs. Finally, the number of stacked layers in the BERT Bi-Stacked-LSTM model was set to 2. An overview of all the parameters is presented in table 6 below.

Table 5: Total number of sentences contained in each dataloader

NUMBER OF SENTENCES IN DATALOADERS
Train DataLoader: 4.190
Validation DataLoader: 1.048
OLID test 2019 DataLoader: 72
OLID test 2020 DataLoader: 324
Combined test DataLoader: 396

Table 6: Hyperparameters set for the proposed model architectures

PARAMETERS OF THE PROPOSED METHODS
Loss Function: Binary Cross-Entropy Loss with Logits
Optimizer: AdamW
Learning Rate: 2e-5 (0.00002)
AdamW Epsilon: 1e-2 (0.01)
Betas: 0.9, 0.999
Batch Size: 12
Number of epochs: 10
Weight Decay: 1e-2 (0.01)
Warm-up Steps: 0
Gradient Clipping Value: 1.0
Dropout: 0.1
Early Stopping Patience: 3
Number of layers in the Bi-Stacked-LSTM: 2

Table 7: The total number of parameters and the total amount of training and validation time of the proposed models

TOTAL NUMBER OF MODEL PARAMETERS	
BERT CLASSIFIER LAST HIDDEN	335.145.985
BERT CLASSIFIER CONCAT LAST 4 HIDDEN	335.149.057
BERT BI-STACKED-LSTM	339.957.041
TOTAL AMOUNT OF TRAINING AND VALIDATION TIME	
BERT CLASSIFIER LAST HIDDEN	13:28:52
BERT CLASSIFIER CONCAT LAST 4 HIDDEN	13:46:55
BERT BI-STACKED-LSTM	14:19:04

4.2.3. Evaluation Metrics

The machine learning models were analyzed in terms of eight standard functional metrics, namely accuracy, precision, recall and F-measure, Mathews Correlation Coefficient (MCC), Confusion Matrix, ROC Curve as well as learning curves of training and evaluation loss and accuracy.

Learning curves: The learning curves are plots that are used to diagnose the optimization of hyperparameters or the performance of a machine learning algorithm and are divided into optimization and performance learning curves respectively. The optimization learning curves usually show the training and validation loss, while the most common performance learning curve illustrates the training and validation accuracy. It is important to examine a model's learning curves, especially its loss learning curves, in order to diagnose and solve problems concerning learning, such as underfitting and overfitting, as well as problems regarding the representativeness of the training and validation sets.

A model can be diagnosed with underfitting when it cannot adequately learn the training set

in order to reach a sufficiently low training error value. Underfitting is apparent in two cases. In the first case, the training learning curve constitutes a flat line or noise values of relatively high loss, demonstrating that the model does not possess the appropriate capacity for the complexity of the dataset. In the second case, the curves of the training and validation loss continue to decrease till the end of the plot, indicating that the model was capable of further learning and that the training process was halted prematurely. Several solutions to underfitting may be to increase the number of training epochs, add early stopping callback and increase the learning rate. A model can be diagnosed with overfitting when it was able to learn the training set too well to generalize on new unseen data, demonstrating statistical noise or random fluctuations. Overfitting is evident when the training loss continues to decrease with experience until the end of the plot, while the validation loss has decreased to a minimum, but has begun increasing. Some ways to deal with overfitting would be to reduce the learning rate, add early stopping callback, add dropout layer, reduce the model capacity (for example the number or size of hidden layers) and regularize the weights. On the other hand, a good model fit is considered when the curve of the training loss is lower than the curve of the validation loss, they both have reached to a point of stability and there is a small generalization gap between them. If a good fit model continues to train, it is likely to overfit.

From the learning curves, it can also be observed whether the training or validation sets possess the appropriate amount of data or adequate information. In other words, a training set is considered unrepresentative when it cannot provide the model with sufficient information to learn, either because it consists of a small number of examples or because it contains features with less variance than the validation set. The unrepresentativeness of the training set can be detected when both training and validation loss decrease, whereas there is a large gap between them. Finally, a validation set is deemed unrepresentative when it fails to provide the model with the ability to generalize, since it contains a much smaller number of examples compared to the training set. The unrepresentativeness of the validation set can be identified when the training loss has decreased and reached a plateau, while the validation loss shows noisy fluctuations. In the present thesis, both the learning curves of loss and accuracy are presented and observed.

Accuracy: The accuracy describes how the model performs across all classes. It is calculated as the ratio between the number of correct predictions to the total number of predictions (true values of examples). It is mainly useful when the number of examples in all classes are equal. In case of class imbalance, it is very likely that the accuracy for the majority class will be higher than the accuracy for the rest of the classes.

Precision, Recall, F1 score: The precision measures the ability of the model to classify all positive examples as Positive and not misclassify a negative sample as Positive. It is calculated as the ratio between the number of Positive examples correctly classified to the total number of examples classified as Positive.

The recall measures the ability of the model to detect the Positive examples. It is calculated

as the ratio between the number of Positive examples that were correctly classified as Positive to the total number of Positive examples. It does not take into account how the negative examples are classified. In case the recall is too high, it denotes that the model has successfully detected all the positive examples.

While it is significant for a model to have both high precision and recall, this is not always the case in real-life situations. A machine learning engineer would have to deal with the so-called Precision-Recall Trade-Off, as the precision calculates the extent of error caused by False Positives (FPs), whereas the recall calculates the extent of error caused by False Negatives (FNs). A model can be regularized to achieve high score in one metric at the cost of achieving lower score in the other. For this reason, the F1 score was created since it constitutes the combination of precision and recall into a single metric. It is calculated as the harmonic mean of precision and recall and it is considered a more appropriate metric to evaluate imbalanced data compared to accuracy (Korstanje, 2021). For the proposed task, the macro-F1 score, macro-precision and macro-recall are calculated, which are the unweighted-average metrics of both classes.

Confusion Matrix: A confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N denotes the number of target classes. It compares the actual target values with the predicted ones by the model. It provides a comprehensive view of the model's performance and of its types of errors. For a binary classification problem like the one examined in the thesis, a 2 x 2 matrix with four values is created. On the first row of the matrix, the True Positive (TP) on the left and the False Positive (FP) on the right values are shown. The TP denotes that the actual value was positive and the model predicted it as positive, while the FP or Type 1 error means that the actual value was negative, but the model predicted it as positive. On the second row of the matrix, the False Negative (FN) on the left and the True Negative (TN) values are depicted. The FN or Type 2 error means that although the actual value was positive, the model predicted it as negative. Finally, the TN denotes that the actual value was negative and the model predicted it as negative (Bhandari, 2022). The values for the present binary classification task are presented as percentages to aid comprehension.

MCC: The Mathews Correlation Coefficient (MCC) is a metric that calculates the values from a confusion matrix and presents results between -1 and 1. A score of 1 indicates perfect agreement, whereas a score of -1 demonstrates total disagreement between the predicted and actual classes. A score of 0 shows completely random guessing. Chicco et al. (2020), recommend the MCC as a more reliable metric for imbalanced datasets compared to F1 score and accuracy as they can offer misleading results.

AUC - ROC curve: The Area Under Curve (AUC) represents the ability of the model to distinguish between classes. A model that achieves AUC score near 1 means that is good at predicting the predicted class as the actual class, whereas a score near 0 denotes that the model is not able to differentiate a class from another. A score at 0.5 means that the model

cannot distinguish the classes at all (Narkhede, 2021).

The Receiver Operating Characteristic (ROC) curve is a performance measurement for classification problems at various threshold settings. It shows trade-off between sensitivity (or TPR) and specificity (1 – FPR). The True Positive Rate (TPR) is plotted on the y axis, while the False Positive Rate (FPR) on the x axis. The TPR is the ratio of observations that are correctly predicted to be positive out of all positive observations. The FPR is the ratio of observations that are incorrectly predicted to be positive out of all negative observations. A curve that is closer to the top-left corner indicates a better performance (Narkhede, 2021).

4.3. Experimental Results

In the present chapter, the three models are evaluated on the training and validation sets of the balanced dataset based on the learning curves of loss and accuracy that are displayed. Moreover, the results of the proposed deep learning models based on the two test sets are illustrated in tables and their performance is compared. Finally, the performance of those models is compared to the performance of other machine learning models introduced in previous related work.

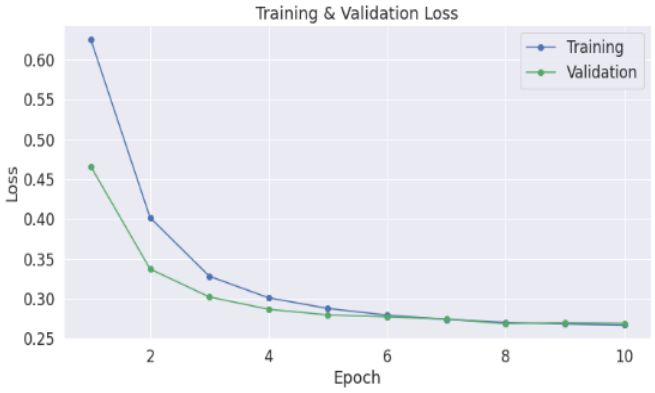
4.3.1. Evaluation and Comparison of Models (Training and Validation Sets)

Optimization Learning Curves of Loss

1. BERT Classifier Last Hidden

2. BERT Classifer Concat Last 4 Hidden

3. BERT Bi-Stacked-LSTM

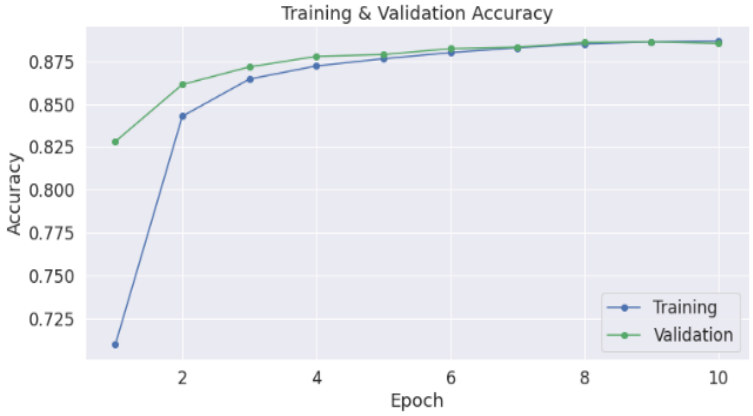


Performance Learning Curves of Accuracy

1. BERT Classifier Last Hidden

2. BERT Classifer Concat Last 4 Hidden

3. BERT Bi-Stacked-LSTM



From the learning curves of loss, it is shown that the three models are a good fit. Although the training loss is very high at the beginning in all models, especially in the BERT Bi-Stacked-LSTM, it decreases steadily to a point of stability. The same happens with the validation loss, which also decreases to a point of stability and has a very small, even non-existent in the BERT Bi-Stacked-LSTM, generalization gap with the training loss. The exceptional performance of all three models on both sets is also evident from the learning curves of accuracy, since the curves increase over time until they reach a point of stability with a very small gap between them. Therefore, it is revealed that the balanced dataset contributed to good model training and model performance, as the models seem to have learned well the training data and are capable of generalization; making correct predictions on new unseen data.

4.3.2. Evaluation and Comparison of Proposed Models (Test Sets)

Table 8: Performance of models based on metrics

OLID TEST 2019 SCORES			
METRICS	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
F1 Score	75.30%	**77.80%**	74%
Accuracy	81.10%	**82.40%**	80.10%
MCC	51.10%	**55.70%**	48.50%
ROC AUC	74%	**77%**	73%
OLID TEST 2020 SCORES			
METRICS	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
F1 Score	85.60%	**86.80%**	84.50%
Accuracy	88.70%	**89.50%**	87.90%
MCC	71.30%	**73.70%**	69.30%
ROC AUC	85%	**87%**	84%
COMBINED OLID TESTS SCORES			
METRICS	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
F1 Score	83.70%	**85.20%**	82.70%
Accuracy	87.30%	**88.20%**	86.50%
MCC	67.70%	**70.40%**	65.60%
ROC AUC	83%	**85%**	82%

Taking into consideration the tables above, it is demonstrated that the best performing model on the balanced dataset is the BERT Classifier with the concatenation of the last four hidden states. It achieved a macro-F1 score of 77.8%, 86.8% and 85.2% on the OLID 2019, OLID 2020 and combined OLID test sets respectively. Its MCC scores are closer to 100%, which indicates that there is relatively good agreement between the predicted and the actual classes. Its ROC AUC scores are also higher than the other models in all test sets, which denotes that the model is able to differentiate between the classes and predict correctly. As far as the classes are concerned, the BERT Classifier Concat Last 4 Hidden achieved an F1 score of 67.8% on the offensive and 87.9% on the not offensive class on test OLID 2019, whereas it achieved an F1 score of 80.9% on the offensive and 92.8% on the not offensive class on test OLID 2020. The concatenation of the two test sets resulted in 78.5% F1 score for the offensive and 91.9% for the not offensive class.

Second in place comes the BERT Classifier by taking only the last hidden state. On test OLID 2019, it achieved an F1 score of 63.3% and 87.3% on the offensive and not offensive class respectively. However, it achieved an F1 score of 78.8% on the offensive and 92.3% on the not offensive class on test OLID 2020. The combined test set resulted in 76.1% F1 score for the offensive and 91.4% for the not offensive class.

Interestingly enough, the BERT Bi-Stacked-LSTM model received the final place, as it performed the lowest among the three models. Even though it can be observed that the model

was a very good fit and performed exceptionally well based on the learning curves, the results on the test sets were not as good as expected. When I decided to develop such a model, I hoped that it would prove to be by far the best. Nevertheless, it seems that a simpler in architecture model managed to make more successful predictions than the complex with much more parameters model. On test OLID 2019, it achieved an F1 score of 61.5% on the offensive and 86.5% on the not offensive class. On test OLID 2020, it achieved an F1 score of 77.3% and 91.7% on the offensive and not offensive class respectively. The concatenation of the two test sets resulted in 74.5% F1 score for the offensive and in 90.8% for the not offensive class. An overview of the F1 score results for each class is presented on table 8.

At this point, it is important to consider the discrepancy of class distribution between the training, validation and test sets as well as between the test sets. That is to say, the number of tweets in the test sets is much lower than in the training and validation sets, hence, offering lower results than expected. Apart from this, the results of OLID 2019 are lower than the results of OLID 2020 test set, especially on the offensive class, due to the fact that the first test set contains 3.027 tweets less than the latter, while its offensive class consists the complete minority, as it includes 380 tweets less than its not offensive class. In this way, a difficulty is added to the task, however, variance of class distribution is a common situation in real-world problems. The classification reports of the three models in each test set are presented in the Appendix.

Table 9: Performance of models on each class based on F1 score

OLID 2019 TEST F1 SCORES			
CLASSES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
Offensive (1)	63.30%	**67.80%**	61.50%
Not Offensive (0)	87.30%	**87.90%**	86.50%
OLID 2020 TEST F1 SCORES			
CLASSES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
Offensive (1)	78.80%	**80.90%**	77.30%
Not Offensive (0)	92.30%	**92.80%**	91.70%
COMBINED OLID TESTS F1 SCORES			
CLASSES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
Offensive (1)	76.10%	**78.50%**	74.50%
Not Offensive (0)	91.40%	**91.90%**	90.80%

Table 10: Performance of models based on confusion matrices in percentages

CONFUSION MATRICES OF OLID TEST 2019			
VALUES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
True Positive (TP)	84.80%	**87.16%**	84.27%
False Positive (FP)	30.69%	30.57%	33.17%
False Negative (FN)	15.20%	12.84%	15.73%
True Negative (TN)	69.31%	**69.43%**	66.83%
CONFUSION MATRICES OF OLID TEST 2020			
VALUES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
True Positive (TP)	90.92%	**92.26%**	90.34%
False Positive (FP)	17.56%	17.78%	19.03%
False Negative (FN)	9.08%	7.74%	9.66%
True Negative (TN)	**82.44%**	82.22%	80.97%
CONFUSION MATRICES OF COMBINED OLID TESTS			
VALUES	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
True Positive (TP)	89.79%	**91.33%**	89.22%
False Positive (FP)	19.78%	20.08%	21.46%
False Negative (FN)	10.21%	8.67%	10.78%
True Negative (TN)	**80.22%**	79.92%	78.54%

From table 10, which illustrates the results from the confusion matrices, it is revealed that BERT Classifier Concat 4 Last Hidden is able to classify correctly more truly offensive tweets as offensive (True Positive) and more truly not offensive tweets as not offensive (True Negative) in the OLID 2019 test set compared to the other models. In OLID 2020 test, while it seems that BERT Classifier Concat 4 Last Hidden can classify a higher percentage of not offensive tweets as actually not offensive (TP), BERT Classifier Last Hidden can predict more offensive tweets as actually being offensive (TN). From the combined OLID test sets, it is evident that BERT Classifier Last Hidden has the potential to predict more offensive tweets as truly offensive than the other models. While the BERT BI-STACKED-LSTM provided satisfactory results, it performed worse compared to the other two models. The confusion matrices as well as the ROC curve of the three models in each test dataset are presented in the Appendix.

4.3.3. Comparison of Models with Previous Related Models

Table 11: Performance of models based on macro-F1 score

OLID TEST 2019 SCORES				
METRIC	**SEMEVAL-2019 SUB-TASK A** ***NULI* TEAM**	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
Macro-F1 score	82.86%	75.30%	77.80%	74%
OLID TEST 2020 SCORES				
METRIC	**SEMEVAL-2020 SUB-TASK A** ***UHH-LT* TEAM**	**BERT CLASSIFIER LAST HIDDEN**	**BERT CLASSIFIER CONCAT LAST 4 HIDDEN**	**BERT BI-STACKED-LSTM**
Macro-F1 score	92.04%	85.60%	86.80%	84.50%

In task 6 of the *SEMEVAL-2019 (OFFENSEVAL)* competition, 104 teams participated in sub-task A, *Offensive Language Detection*. It was a binary classification task consisting of the classes OFF and NOT. The goal and classes of the present thesis were in line with this competition. According to Zampieri et al. (2019), seven out of the top ten teams employed BERT with different parameters and pre-processing steps. The best performing team on this sub-task that achieved the first place was the *NULI* with a macro-averaged F1 score of 82.86% and accuracy 86.28%. The *NULI* team fine-tuned BERT-Base-Uncased for 3 epochs using default parameters and a maximum sequence length of 64. They applied a number of pre-processing techniques, like hashtag segmentation and emoji substitution (Liu et al., 2019).

In task 12 of the competition that took place next year, the *SEMEVAL-2020 (OFFENSEVAL)* competition, 84 teams participated in the extension of sub-task A (Zampieri et al,. 2020). The *UHH-LT* TEAM won the first place with a macro-averaged F1 score of 92.04%. They developed an MLM pre-trained RoBERTa ensemble model, which was fine-tuned 10 times for 6 epochs with learning rate 5e-6, batch size 8 and maximum sequence length of 128. They aggregated the final predictions from each model and created an ensemble model by using the majority vote approach (Wiedemann et al., 2020).

From the tables above, it is clearly illustrated that the macro-F1 scores achieved by the submission models of the teams in the competitions outperform all three models introduced in the present thesis. One must certainly take into consideration the different training sets used to train the models and the different model hyperparameters, not to mention the fact the *UHH-LT* TEAM utilized an ensemble of many models in order to achieve such a F1 score, while single models were introduced and trained in this thesis.

5. LIMITATIONS & CONCLUSION

Considering the prevalence of offensive language in social media and its negative impact on the society, the development of machine learning models that have the potential to distinguish between offensive and not offensive language is of paramount importance. Even though governments, social media companies and the academic community have made concerted efforts, offensive language detection poses a great challenging issue nowadays. Offensive language is usually difficult for a machine learning system to identify, as it does not possess knowledge of several linguistic features, like ways expressing irony, hate or sarcasm as well as idiomatic expressions.

In this thesis, three machine learning methods based on BERT, the pre-trained Transformer model, were introduced for offensive language detection in English tweets. The effectiveness of the methods was evaluated on six metrics and the learning curves of loss and accuracy. The results illustrated that utilizing the contextual representations from different layers of a Transformer can improve the performance of a model depending on the task, since the concatenation of the last four hidden layers in the BERT Classifier model outperformed the BERT Classifier with the last hidden layer and the BERT Bi-Stacked-LSTM models explored in this study. It was also revealed that the simpler in architecture classification models could achieve higher results than the more complex model. A thoughtful analysis would conclude that the proposed methods, especially the best performing model, can be employed to classify offensive language efficiently in many tweets, in spite of being my first attempt in binary text classification using Transformers. In case certain adjustments and upgrades are being made, the effectiveness of the models might improve to the point of reaching the extremely high results achieved from the competition models. Although there were certain restrictions concerning the imbalance of available datasets, the limited GPU capacity as well as my limited experience in the NLP and machine learning field, it can be considered that the present paper provides an intuitive approach towards text classification and machine learning methods and contributes to the NLP community with this task.

6. FUTURE WORK

Even though the three proposed model architectures based on BERT offered satisfactory results, different lines of research may be explored in the future. To begin with, more English data can be collected, mainly including offensive tweets, with a view to creating a sizeable dataset while handling the class imbalance between offensive and not offensive tweets. Without a doubt, a large and balanced dataset that could be provided for free to the NLP community would contribute to offensive language detection, since models in future studies would train on more data without any bias towards a specific class. Apart from this, a good idea for future research would be to consider the whole context of a tweet including the re-twitting and chain discussions. It may also be useful to target several users, who regularly post offensive tweets and collect their posts for further examination. Moreover, the binary classification might be expanded to multi-label classification, so that offensive language is differentiated into sub-categories, for instance *hate*, *sexist*, *racist*, *sexual harassment*, *body shamming*, that can facilitate detection and consequent elimination of such incidents occurring online. Additionally, since offensive language cannot only be expressed in English, the study can be extended to investigate other languages as well. It has come to my attention that limited research has been conducted in my native language. Hence, in future work, I plan to explore multi-lingual pre-trained models for offensive language detection in Greek. Furthermore, other pre-trained Transformer models and architectures, like the ones developed for the competitions, can be utilized for the task in order to achieve higher results. Experimenting with different pre-processing steps might lead to greater performance as well. Last but not least, a future direction of this research would be to retrieve two multi-modal sources of information, text and image, not only from Twitter, but also from different social media platforms and combine them to improve the classification task.

APPENDIX

Loss and Accuracy of models on training and validation sets

In this part, the results of loss and accuracy for each epoch as well as the training and validation time are presented in tables for each model.

BERT Classifier Last Hidden

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Training Time	Validation Time
1	0.44	0.31	0.79	0.85	01:15:04	00:05:49
2	0.30	0.28	0.86	0.87	01:14:58	00:05:49
3	0.28	0.27	0.87	0.88	01:15:01	00:05:49
4	0.27	0.26	0.88	0.89	01:14:58	00:05:49
5	0.26	0.26	0.88	0.89	01:14:57	00:05:49
6	0.26	0.25	0.89	0.89	01:15:03	00:05:50
7	0.25	0.25	0.89	0.89	01:15:00	00:05:49
8	0.25	0.25	0.89	0.89	01:14:59	00:05:49
9	0.25	0.25	0.89	0.89	01:15:02	00:05:49
10	0.25	0.25	0.89	0.89	01:15:01	00:05:50

BERT Classifier Concat Last 4 Hidden

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Training Time	Validation Time
1	0.40	0.30	0.81	0.86	01:16:48	00:05:51
2	0.29	0.27	0.87	0.88	01:16:47	00:05:51
3	0.27	0.26	0.88	0.89	01:16:47	00:05:51
4	0.26	0.25	0.88	0.89	01:16:44	00:05:51
5	0.26	0.25	0.89	0.89	01:16:47	00:05:51
6	0.25	0.25	0.89	0.89	01:16:48	00:05:51
7	0.25	0.25	0.89	0.89	01:16:39	00:05:51
8	0.25	0.25	0.89	0.89	01:16:41	00:05:51
9	0.25	0.25	0.89	0.89	01:16:42	00:05:51
10	0.24	0.25	0.89	0.89	01:16:50	00:05:51

BERT Bi-Stacked-LSTM

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Training Time	Validation Time
1	0.63	0.47	0.71	0.83	01:19:57	00:05:54
2	0.40	0.34	0.84	0.86	01:19:58	00:05:55
3	0.33	0.30	0.86	0.87	01:19:56	00:05:54
4	0.30	0.29	0.87	0.88	01:19:56	00:05:54
5	0.29	0.28	0.88	0.88	01:19:56	00:05:54
6	0.28	0.28	0.88	0.88	01:19:56	00:05:54
7	0.27	0.27	0.88	0.88	01:19:55	00:05:54
8	0.27	0.27	0.89	0.89	01:19:55	00:05:54
9	0.27	0.27	0.89	0.89	01:19:59	00:05:55
10	0.27	0.27	0.89	0.89	01:20:00	00:05:55

Classification Reports of Models

BERT Classifier Last Hidden

OLID 2019 Test Set

	precision	recall	f1-score	support
NOT	0.85	0.90	0.87	620
OFF	0.69	0.58	0.63	240
accuracy			0.81	860
macro avg	0.77	0.74	0.75	860
weighted avg	0.80	0.81	0.81	860

OLID 2020 Test Set

	precision	recall	f1-score	support
NOT	0.91	0.94	0.92	2807
OFF	0.82	0.76	0.79	1080
accuracy			0.89	3887
macro avg	0.87	0.85	0.86	3887
weighted avg	0.89	0.89	0.89	3887

Combined OLID Test Sets

	precision	recall	f1-score	support
NOT	0.90	0.93	0.91	3427
OFF	0.80	0.72	0.76	1320
accuracy			0.87	4747
macro avg	0.85	0.83	0.84	4747
weighted avg	0.87	0.87	0.87	4747

BERT Classifier Concat Last 4 Hidden

OLID 2019 Test Set

	precision	recall	f1-score	support
NOT	0.87	0.89	0.88	620
OFF	0.69	0.66	0.68	240
accuracy			0.82	860
macro avg	0.78	0.77	0.78	860
weighted avg	0.82	0.82	0.82	860

OLID 2020 Test Set

	precision	recall	f1-score	support
NOT	0.92	0.93	0.93	2807
OFF	0.82	0.80	0.81	1080
accuracy			0.90	3887
macro avg	0.87	0.87	0.87	3887
weighted avg	0.89	0.90	0.90	3887

Combined OLID Test Sets

	precision	recall	f1-score	support
NOT	0.91	0.93	0.92	3427
OFF	0.80	0.77	0.79	1320
accuracy			0.88	4747
macro avg	0.86	0.85	0.85	4747
weighted avg	0.88	0.88	0.88	4747

BERT Bi-Stacked-LSTM

OLID 2019 Test Set

	precision	recall	f1-score	support
NOT	0.84	0.89	0.87	620
OFF	0.67	0.57	0.62	240
accuracy			0.80	860
macro avg	0.76	0.73	0.74	860
weighted avg	0.79	0.80	0.80	860

OLID 2020 Test Set

	precision	recall	f1-score	support
NOT	0.90	0.93	0.92	2807
OFF	0.81	0.74	0.77	1080
accuracy			0.88	3887
macro avg	0.86	0.84	0.85	3887
weighted avg	0.88	0.88	0.88	3887

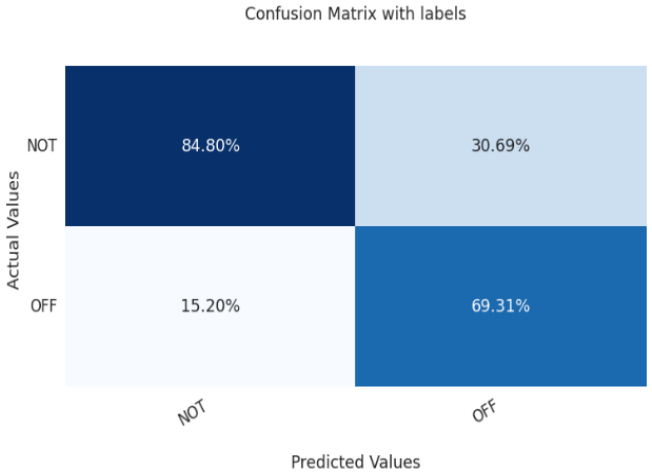
Combined OLID Test Sets

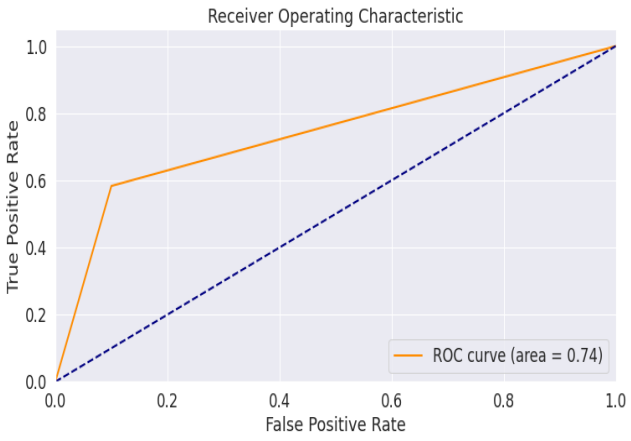
	precision	recall	f1-score	support
NOT	0.89	0.93	0.91	3427
OFF	0.79	0.71	0.75	1320
accuracy			0.87	4747
macro avg	0.84	0.82	0.83	4747
weighted avg	0.86	0.87	0.86	4747

Confusion Matrices and ROC Curves

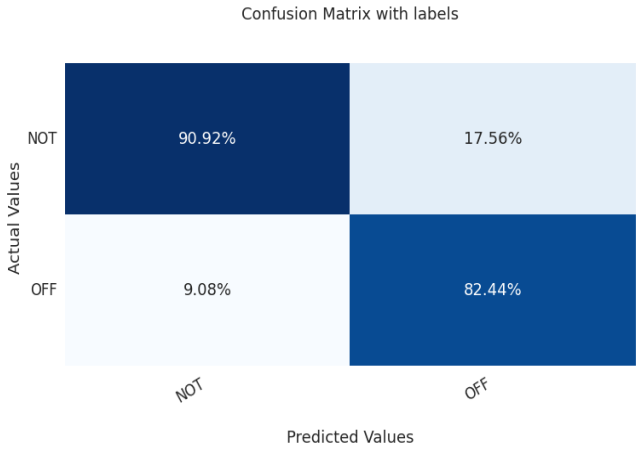
BERT Classifier Last Hidden

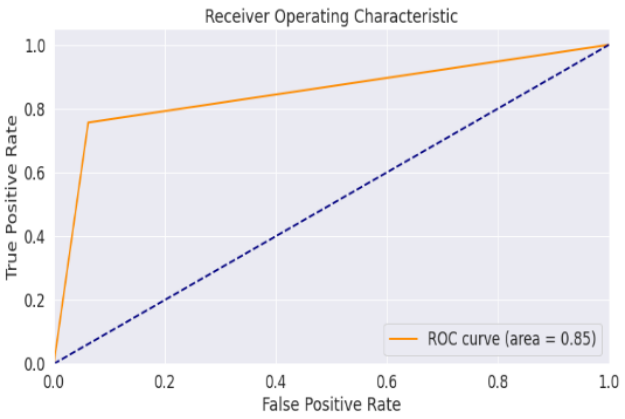
OLID 2019 Test Set



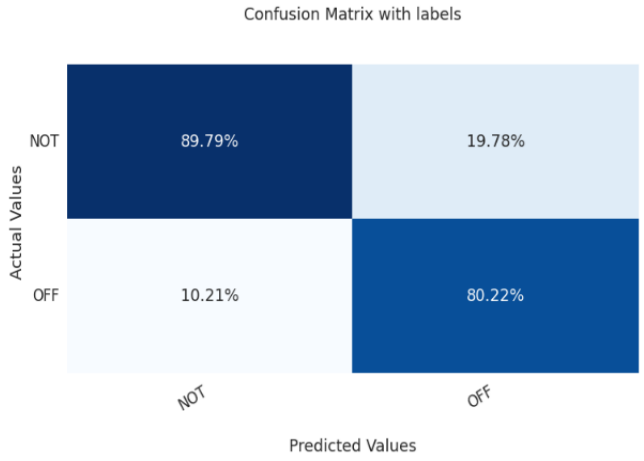


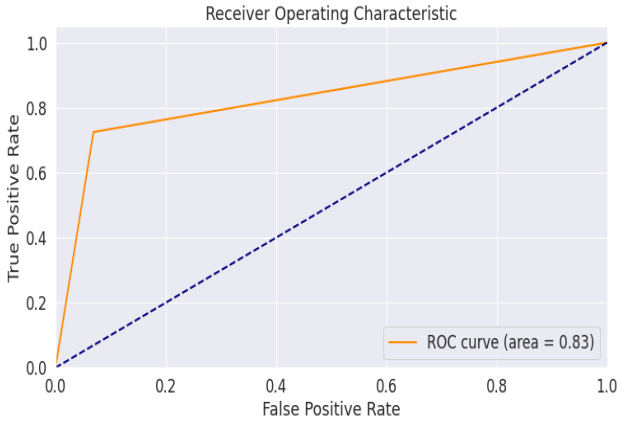
OLID 2020 Test Set





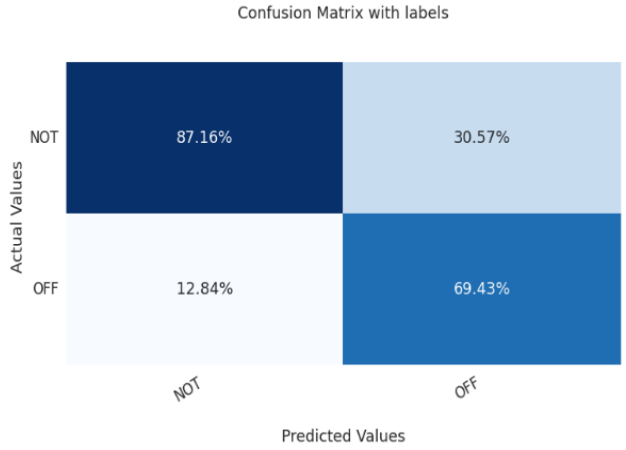
Combined OLID Test Sets

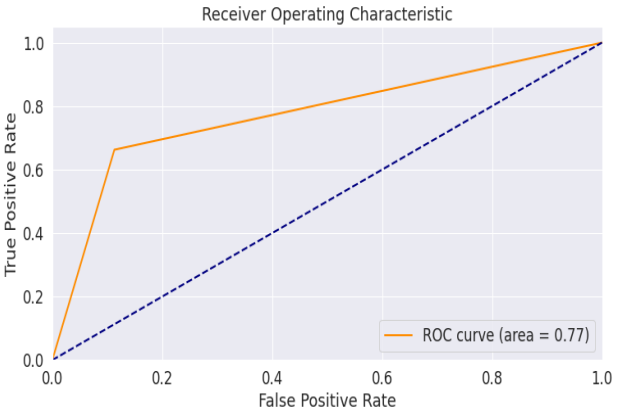




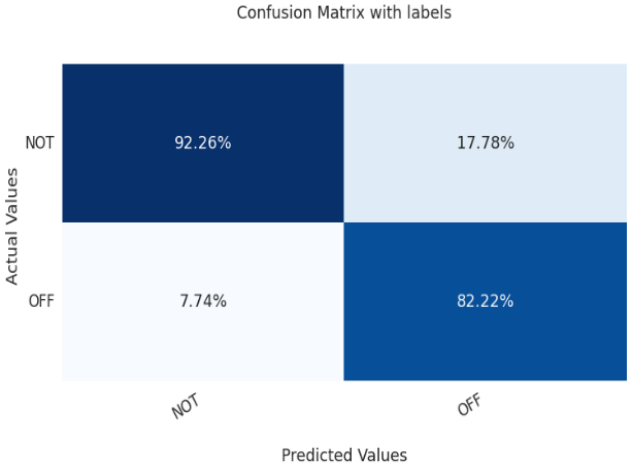
BERT Classifier Concat Last 4 Hidden

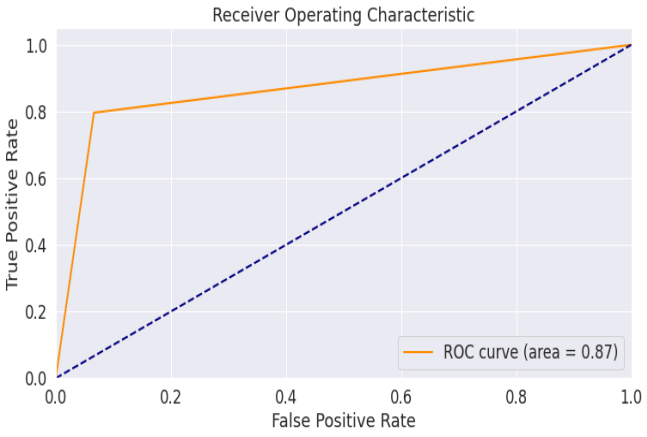
OLID 2019 Test Set



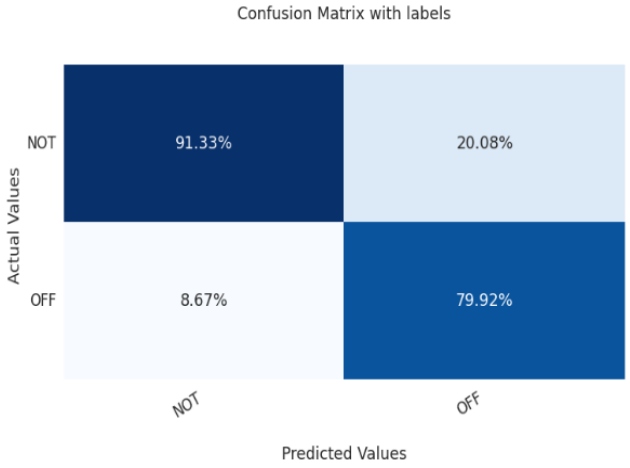


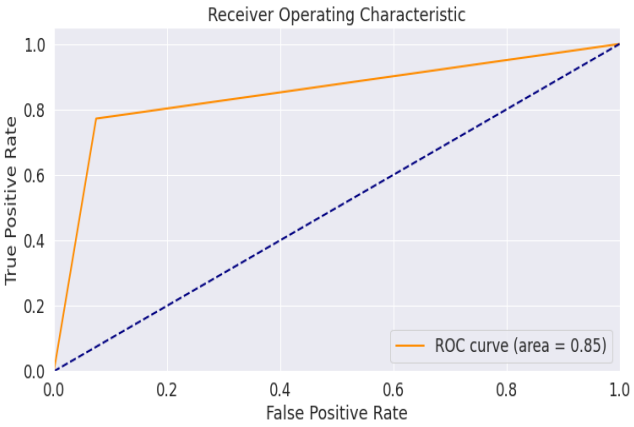
OLID 2020 Test Set





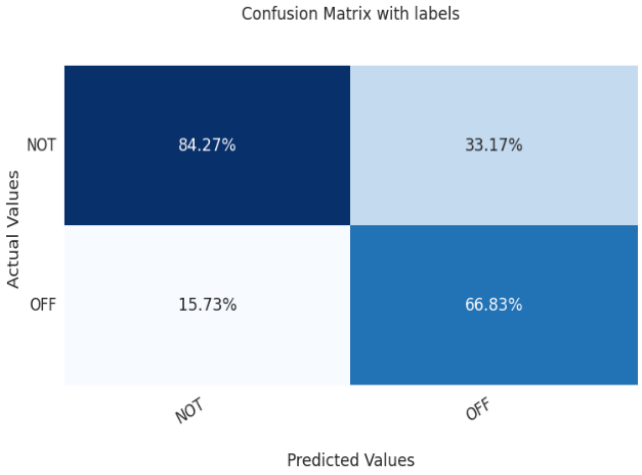
Combined OLID Test Sets

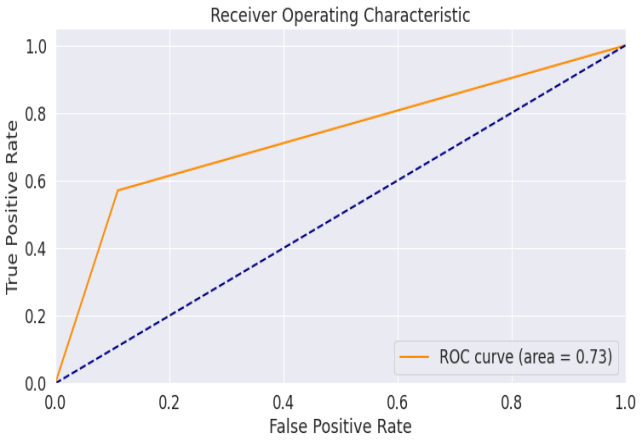




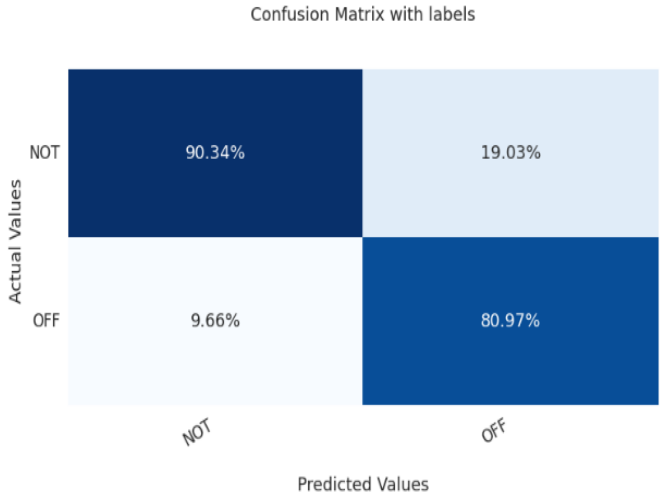
BERT Bi-Stacked-LSTM

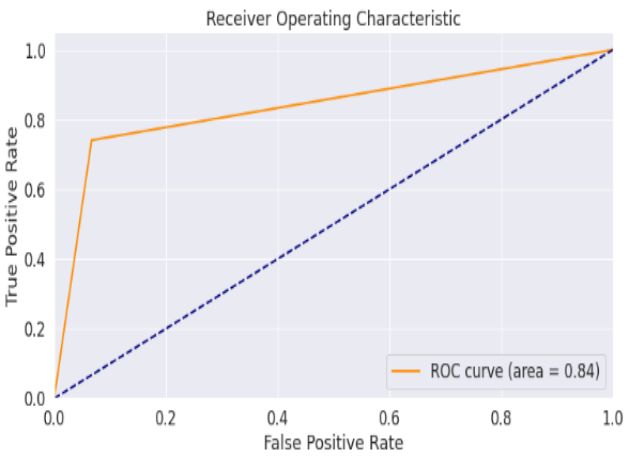
OLID 2019 Test Set



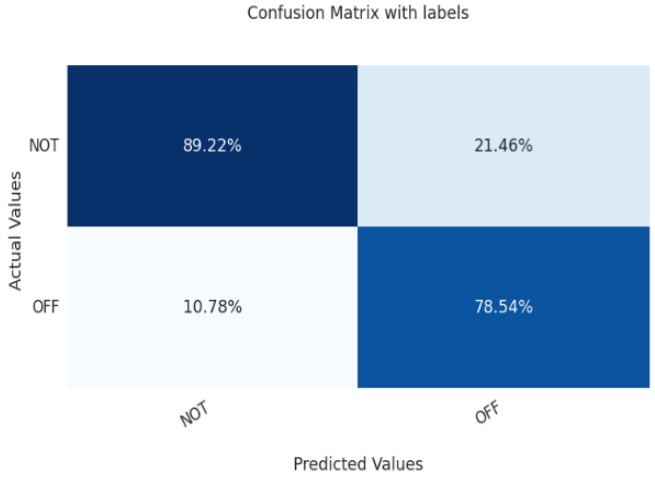


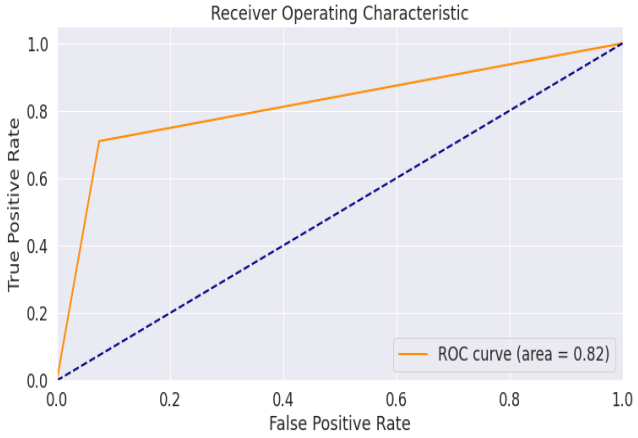
OLID 2020 Test Set





Combined OLID Test Sets





REFERENCES

1. Akhter, M. P., Jiangbin, Z., Naqvi, I. R., AbdelMajeed, M., & Zia, T. (2021). Abusive language detection from social media comments using conventional machine learning and deep learning approaches. *Multimedia Systems*. https://doi.org/10.1007/s00530-021-00784-8
2. Alammar, J. (2018, December 3). The illustrated Bert, Elmo, and Co. (how NLP cracked transfer learning). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Jay Alammar – Visualizing machine learning one concept at a time. Retrieved September 23, 2022, from http://jalammar.github.io/illustrated-bert/
3. Aragon, M. E., Carmona, M. A. A., Montes, M., & Escalante, H. J. (2019, August). *Overview of MEX-A3T at IberLEF 2019: Authorship and aggressiveness analysis in Mexican Spanish tweets*. Notebook Papers of 1st SEPLN Workshop on Iberian Languages Evaluation Forum (IberLEF), Bilbao, Spain.
4. Aroyehun, S., T., & Gelbukh, A. (2018). Aggression Detection in Social Media: Using Deep Neural Networks, Data Augmentation, and Pseudo Labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97, Santa Fe, New Mexico, USA. Association for Computational Linguistics. Retrieved October 1, 2022, from https://aclanthology.org/W18-4411
5. Arsht, A., & Etcovitch, D. (2018, March 2). *The Human Cost of Online Content Moderation*. Harvard Journal of Law & Technology. Retrieved August 23, 2022, from https://jolt.law.harvard.edu/digest/the-human-cost-of-online-content-moderation
6. Baggs, M. (2021, November 15). *Online hate speech rose 20% during pandemic: 'we've Normalised it'*. BBC News. Retrieved August 12, 2022, from https://www.bbc.com/news/newsbeat-59292509
7. Barbieri, F., Camacho-Collados, J., Espinosa Anke, L., & Neves, L. (2020). Tweeteval: Unified benchmark and comparative evaluation for Tweet Classification. *Findings of the Association for Computational Linguistics: EMNLP 2020*. https://doi.org/10.18653/v1/2020.findings-emnlp.148
8. Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., Rosso, P., & Sanguinetti, M. (2019). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. 54-63. https://doi.org/10.18653/v1/s19-2007
9. Baziotis, C., Pelekis, N., & Doulkeridis, C. (2017). Datastories at Semeval-2017 Task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. https://doi.org/10.18653/v1/s17-2126
10. BBC. (2016, May 31). *Web firms pledge to tackle online hate speech*. BBC News. Retrieved August 14, 2022, from https://www.bbc.com/news/technology-36416967
11. Bhandari, A. (2022, June 14). *Confusion matrix for Machine Learning*. Analytics Vidhya. Retrieved September 16, 2022, from https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/
12. Cachola, I., Holgate, E., Preoţiuc-Pietro, D., & Li, J. J. (2018, August). *Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media*. ACL Anthology. 2927–2938. Retrieved September 14, 2022, from https://aclanthology.org/C18-1248
13. Cain, A. C., & Gonzalez, B. (2017, January 13). *Opinion | Twitter Must Do More to Block ISIS*. The New York Times. Retrieved August, 17, 2022 from https://www.nytimes.com/2017/01/13/opinion/twitter-must-do-more-to-block-isis.html
14. Chaudhary, M., Saxena, C., & Meng, H. (2021, September 7). *Countering Online Hate Speech: An NLP Perspective*. https://arxiv.org/abs/2109.02941
15. Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1). https://doi.org/10.1186/s12864-019-6413-7
16. Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). *Automated hate speech detection and the problem of offensive language*. Proceedings of the Eleventh International AAAI Conference on Web and Social Media (ICWSM 2017). Retrieved September 12, 2022, from https://arxiv.org/abs/1703.04009

17. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved September 10, 2022, from https://doi.org/10.48550/arXiv.1810.04805
18. Diagnosing model performance with learning curves. (n.d.). Retrieved September 16, 2022, from https://rstudio-conf-2020.github.io/dl-keras-tf/notebooks/learning-curve-diagnostics.nb.html
19. European Commission (2022). *The Digital Services Act package | Shaping Europe's digital future*.Digital-Strategy.ec.europa.eu. Retrieved August 25, from https://digital-strategy.ec.europa.eu/en/policies/digital-services-act-package
20. GOV.UK (2022, April 19). Online Safety Bill: factsheet. Retrieved August 14, 2022, from https://www.gov.uk/government/publications/online-safety-bill-supporting-documents/online-safety-bill-factsheet
21. Goldman, R. (2010, January 26). *Teens indicted after allegedly taunting girl who hanged herself*. ABC News. Retrieved August 12, 2022, from https://abcnews.go.com/Technology/TheLaw/teens-charged-bullying-mass-girl kill/story?id=10231357
22. Hern, A. (Ed.). (2021, May 6). *Twitter launches prompt asking users to rethink abusive tweets*. The Guardian. Retrieved August 17, 2022, from https://www.theguardian.com/technology/2021/may/06/twitter-launches-prompt-in-bid-to-reduce-abusive-language
23. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
24. Islam, M. M., Uddin, M. A., Islam, L., Akter, A., Sharmin, S., & Acharjee, U. K. (2020). Cyberbullying detection on social networks using machine learning approaches. *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. https://doi.org/10.1109/csde50874.2020.9411601
25. Janardhana, D. R., Shetty, A. B., Hegde, M. N., Kanchan, J., & Hegde, A. (2020). Abusive comments classification in social media using Neural Networks. *Advances in Intelligent Systems and Computing*, 439–444. https://doi.org/10.1007/978-981-15-5113-0_33
26. Khalid, S. (2020, April 10). *Bert explained: A Complete Guide with Theory and tutorial*. Medium. Retrieved September 11, 2022, from https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c
27. Korstanje, J. (2021, August 31). *The F1 score*. Medium. Retrieved September 16, 2022, from https://towardsdatascience.com/the-f1-score-bec2bbc38aa6
28. Kumar, R., Bhanodai, G., Pamula, R., & Chennuru M., R. (2018). TRAC-1 Shared Task on Aggression Identification: IIT(ISM)@COLING'18. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 58–65, Santa Fe, New Mexico, USA. Association for Computational Linguistics. Retrieved October 1, 2022, from https://aclanthology.org/W18-4407
29. Liu, P., Li, W., & Zou, L. (2019). Nuli at Semeval-2019 Task 6: Transfer Learning for offensive language detection using bidirectional transformers. *Proceedings of the 13th International Workshop on Semantic Evaluation*. https://doi.org/10.18653/v1/s19-2011
30. Modha, S., Majumder, P., & Mandl, T. (2018). Filtering Aggression from the Multilingual Social Media Feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 199–207, Santa Fe, New Mexico, USA. Association for Computational Linguistics. Retrieved October 1, 2022, from https://aclanthology.org/W18-4423
31. Modinos, G. (2022, May 18). *New Digital Services Act*. Lambadarios. Retrieved August 13, 2022, from https://www.lambadarioslaw.gr/2022/05/new-digital-services-act/
32. Mutanga, R. T., Naicker, N., & Olugbara, O. O. (2020, August 30). *Hate speech detection in Twitter using Transformer methods*. International Journal of Advanced Computer Science and Applications (IJACSA). Retrieved September 11, 2022, from http://dx.doi.org/10.14569/IJACSA.2020.0110972
33. Narkhede, S. (2021, June 15). *Understanding AUC - roc curve*. Medium. Retrieved September 16, 2022, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
34. Oblad, T. (2019, October 1). *Social media use, depression, and self-harm among youth*. Institute for Family Studies. Retrieved August 13, 2022, from https://ifstudies.org/blog/social-media-use-depression-and-self-harm-among-youth

35. Okky I., M., Sazany, E., & Budi, I. (2019). Identify Abusive and Offensive Language in Indonesian Twitter using Deep Learning Approach. *Journal of Physics: Conference Series, 1196*, 012041. https://doi.org/10.1088/1742-6596/1196/1/012041
36. *Online hate crime*. Stop Hate UK. (2021, November 24). Retrieved August 12, 2022, from https://www.stophateuk.org/about-hate-crime/what-is-online-hate-crime/
37. *Online safety bill: Factsheet*. GOV.UK. (2022, April 19). Retrieved August 14, 2022, from https://www.gov.uk/government/publications/online-safety-bill-supporting-documents/online-safety-bill-factsheet
38. Pitenis, Z., Zampieri, M., & Ranasinghe, T. (2020). *Offensive Language Identification in Greek*. Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France. European Language Resources Association. 5113–5119. https://aclanthology.org/2020.lrec-1.629
39. Pitsilis, G. K., Ramampiaro, H., & Langseth, H. (2018). Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence*, *48*(12), 4730–4742. https://doi.org/10.1007/s10489-018-1242-y
40. Reynolds, K., Kontostathis, A., & Edwards, L. (2011). Using machine learning to detect cyberbullying. *2011 10th International Conference on Machine Learning and Applications and Workshops*. https://doi.org/10.1109/icmla.2011.152
41. Risch, J., Stoll, A., Wilms, L., & Wiegand, M. (2021). Overview of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments. GERMEVAL. Psychology. Retrieved October 3, 2022, from https://aclanthology.org/2021.germeval-1.1.pdf
42. Saxena, S. (2021, March 18). *Introduction to Long Short Term Memory (LSTM)*. Analytics Vidhya. Retrieved September 22, 2022, from https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/
43. Song, Y., Wang, J., Liang, Z., Liu, Z., & Jiang, T. (2020, February 12). *Utilizing Bert Intermediate layers for aspect based sentiment analysis and natural language inference*. arXiv.org. Retrieved September 23, 2022, from https://doi.org/10.48550/arXiv.2002.04815
44. Sossi Instructional Assistant, D. (2022, January 20). *And just like that ... #MeToo changed the nature of online communication*. The Conversation. Retrieved August 12, 2022, from https://theconversation.com/and-just-like-that-metoo-changed-the-nature-of-online-communication-174527
45. Spertus. E. (1997). Smokey: Automatic recognition of hostile messages. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI/IAAI, pages 1058–1065, Providence, RI, USA. AAAI Press. Retrieved October 1, 2022, from https://www.cs.csustan.edu/~mmartin/LDS/Spertus.pdf
46. Struß, J., M., Siegel, M., Ruppenhofer, J., Wiegand, M. & Klenner, M. (2019). Overview of GermEval Task 2, 2019 Shared Task on the Identification of Offensive Language. In *German Society for Computational Linguistics. Proceedings of the 15th Conference on Natural Language Processing (KONVENS) 2019*. 354-365. https://doi.org/10.5167/UZH-178687
47. Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification? *Lecture Notes in Computer Science*, 194–206. https://doi.org/10.1007/978-3-030-32381-3_16
48. Twitter. (n.d.). *Counting characters | docs | twitter developer platform*. Twitter. Retrieved September 20, 2022, from https://developer.twitter.com/en/docs/counting-characters
49. *USA: Twitter failing to protect pro-choice abortion activists from online violence.* Amnesty International UK. (2022, July 7). Retrieved August 12, 2022, from https://www.amnesty.org.uk/press-releases/usa-twitter-failing-protect-pro-choice-abortion-activists-online-violence
50. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based Natural Language Processing [review article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75. https://doi.org/10.1109/mci.2018.2840738
51. Wakefield, J. (2021, December 6). *Online safety bill : What to expect*. BBC News. Retrieved August 14, 2022, from https://www.bbc.com/news/technology-59509702
52. Wiedemann, G., Yimam, S. M., & Biemann, C. (2020). UHH-lt at Semeval-2020 task 12: Fine-tuning of pre-

trained transformer networks for offensive language detection. *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. https://doi.org/10.18653/v1/2020.semeval-1.213

53. Wiegand, M., Siegel, M., & Ruppenhofer, J. (2018). Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of the GermEval 2018 Workshop (GermEval).* Retrieved October 2, 2022, from https://d-nb.info/1179775287/34
54. Wright, M. F., Harper, B. D., & Wachs, S. (2019). The associations between cyberbullying and callous-unemotional traits among adolescents: The moderating effect of online disinhibition. *Personality and Individual Differences*, *140*, 41–45. https://doi.org/10.1016/j.paid.2018.04.001
55. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., &Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in Social Media (Offenseval). *Proceedings of the 13th International Workshop on Semantic Evaluation*. https://doi.org/10.18653/v1/s19-2010
56. Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., & Çöltekin, Ç. (2020). Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. https://doi.org/10.18653/v1/2020.semeval-1.188
57. Zvornicanin, E. (2022, February 5). *Differences between bidirectional and Unidirectional LSTM*. Baeldung on Computer Science. Retrieved September 21, 2022, from https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm