NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Offensive Language Classification in Social Media

Using Deep Learning

Susan Wei Chen Wang

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

Offensive Language Classification in Social Media Using Deep Learning

by

Susan Wei Chen Wang

Dissertation presented as the partial requirement for obtaining a Master's degree in Data Science and Advanced Analytics

Advisor: Prof. Doutora Zita Alexandra Magalhães Marinho

August 2020

Acknowledgements

I would like to thank my brilliant supervisor, Zita Marinho, for her continual guidance and inspiration. There are many moments of uncertainty during this research and she always inspired me to do my best and provided very thorough feedback. I am particularly thankful for her help during the process of writing my first paper. It was a big learning curve and I could not have done it without her.

I am grateful for the chance to study at University of Nova, Lisbon and particularly for the dedicated staff in my faculty who are extremely knowledgeable and supportive. Especially professors Jorge Mende, Mauro Castelli and Illya Bakurov for their words of encouragement and wisdom.

My deep appreciation also goes to the many professors and researchers I met while attending ACL and EMNLP conferences, who offered their thoughts in this topic and suggested various interesting techniques and perspectives, particularly Professors Kyunghyun Cho, Isabelle Augenstein, Andre Martins and Goncalo Correia.

This work has been supported by Priberam, who I would like to thank for giving me a chance to work in their office, share friendly chats and utilise their resources.

I would like to thank my family, for always being there for me and for understanding my desire to study overseas. Lastly I would like to thank all my friends in Australia and Portugal, for the joy you bring to my life, and particularly Katherine, Francisco, Bhupendra, Ricardo and Teresa for your wonderful support during this time.

Abstract

As social media usage becomes more integrated into our daily lives, the impact of online abuse also becomes more prevalent. Research in the area of Offensive Language Classification are numerous and often occur in parrallel. Offensive Language Identification Dataset (OLID) schema was introduced with the aim of consolidating related tasks by categorising offense into a three-level hierarchy - detection of offensive posts (Level A), distinguishing between targeted and untargeted offenses (Level B) and then identifying the target of the offense (Level C).

This thesis presents our contribution to the Offensive Language Classification Task (English SubTask A) of OffensEval 2020, and a follow-up study of Offense Type Classification (subTask B) and Offense Target Identification (subTask C) of OffensEval 2019. These tasks follow the OLID schema where each level corresponds to an individual subtask.

For subtask A, the dataset is examined in detail and the most uncertain partitions are removed by an under-sampling technique of the training set. We improved model performance by increasing data quality, taking advantage of further offensive language classification datasets. We fine-tuned separate BERT models from individual datasets and experimented with different ensemble approaches including SVMs, Gradient boosting, AdaBoosting and Logistic Regression to achieve a final ensemble classification model that enhanced macro-F1 score. Our best model, an average ensemble of four different Bert models, achieved 11th place out of 82 participants with a macro F1 score of 0.91344 in the English SubTask A.

The dataset for subtask B and C are highly unbalanced, and modification of the classification thresholds improved classifier performance of the minority classes, which in turn improved the overall performance. Again using the BERT architecture, the models achieved macro-F1 scores of 0.71367 for subTask B and 0.643352 for subTask C, equivalent to the 5th and 2nd places in the respective tasks.

We showed that BERT is an effective architecture for offensive language classification and propose further performance gains are possible by improving data quality.

Keywords: Offensive Language, Hate Speech, Toxic Language, Abusive Language,

Social Media, Twitter, BERT, Transformers, Text Classification, NLP, Natural Language Processing, Deep Learning, Ensembles, Imbalanced Dataset ...

Resumo

Conforme o uso da Social Media se torna mais integrado no nosso dia-a-dia, o impacto do abuso online torna-se também mais prevalente. Pesquisas na área de Classificação de Linguagem Ofensiva são numerosas e ocorrem frequentemente em paralelo. O esquema Offensive Language Identification Dataset (OLID) foi introduzido com o objectivo de consolidar tarefas relacionadas com a categorização de ofensas numa hierarquia de três níveis - detecção de posts ofensivos (nível A), distinção entre ofensas directas e indirectas (nível B) e posteriormente a identificação do visado pela ofensa (nível C).

Esta tese apresenta a nossa contribuição à Offensive Language Classification Task (English sub-tarefa A) da OffensEval 2020, e um subsequente estudo de Offense Type Classification (sub-tarefa B) e Offense Target Identification (sub-tarefa C) da OffensEval 2019. Estas tarefas seguem o esquema OLID onde cada nível corresponde a uma tarefa individual.

Para a sub-tarefa A, o conjunto de informação é examinado em detalhe e as partições mais incertas são removidas por uma técnica de sub-amostragem do conjunto de treinamento. Melhoramos também o desempenho ao melhorar a qualidade da informação, aproveitando de conjuntos mais recentes de classificação de linguagem ofensiva.

Ajustamos modelos BERT disjuntos através de conjuntos de informação individuais e experimentamos com diferentes junções incluindo SVMs, Gradient boosting, AdaBoosting e Regressão Logística para alcançar /* um modelo classificação junção final */ que melhorou a pontuação macro-F1. O nosso melhor modelo, uma junção média de quatro modelos Bert diferentes, alcançou o 11º de 82 participantes com uma pontuação macro de 0,91344 na sub-tarefa A de Inglês.

O conjunto de informação para a sub-tarefa B e C são altamente desequilibrados, e modificar os limiares de classificação melhorou o desempenho de classes minoria, que por sua vez melhoraram o desempenho no geral. Novamente usando a arquitectura BERT, os modelos alcançaram pontuações macro-F1 de 0,71367 para a sub-tarefa B e 0.643352 para a sub-tarefa C, equivalente ao 5º e 2º lugares nas tarefas respectivas. Mostrámos que a arquitectura BERT é eficaz para classificação de linguagem ofensiva e propomos que é possível ganhar desempenho através da melhoria da qualidade da informação.

Palavras-chave: Linguagem Ofensiva, Discurso de Odio, Linguagem Tóxica, Linguagem Abusiva, Redes Sociais, Twitter, BERT, Transformadores, Classificação de Texto, NLP, Processamento de Linguagem Natural, Aprendizagem Profunda, Ensembles ...

Contents

List of Figures

List of Tables

Acronyms

BERT	Bidirectional Encoder Representation from Transformer.
CBOW	Continuous Bag-Of-Words.
CLS	Classification.
CNN	Convolutional Neural Network.
DistilBERT	Distilled version of BERT.
ELMo	Embeddings from Language Models.
FC	Fully Connected.
GloVe	Global Vectors for Word Representation.
GPT	Generative Pre-Training.
GRU	Gated Recurrent Units.
LSTM	Long Short-Term Memory.
MLM	Masked Language Model.
MLP	Multilayer Perceptron.
NER	Named Entity Recognition.
NLI	Natural Language Inference.
NLP	Natural Language Processing.
NNLM	Neural Network Language Model.
NSP	Next Sentence Prediction.
OLID	Offensive Language Identification Datatset.

PMI	Pointwise Mutual Information.
RNN	Recurrent Neural Network.
RoBERTa	Robustly Optimized BERT Pretraining Approach.
SOLID	Semi-Supervised Offensive Language Identification Datatset.
SVM	Support Vector Machine.
ULMFit	Universal Language Model Fine-tuning.

Chapter 1

Introduction

1.1 Background and Motivation

As people turn more and more to the online world for their entertainment, social and communication needs, the anonymity offered by the platforms also draw out diverse and sometimes divisive opinions.

As an example, the current US President Donald Trump frequently use Twitter to engage with his followers, posting controversial statements and inflaming sentiments while bypassing the minimal level of civility and scrutiny required by a more traditional media. It is only very recently that Twitter started to assert some form of moderation - in May 2020, Twitter labelled Trump's false claims about mail-in ballots with a "Fact Check" warning[7], and in June 2020, one of Trump's tweets was labelled as with a "abusive behaviour" warning[12].

A popular Japanese reality TV program, Terrace House, came into global attention in June 2020 when one of its stars, Hana Kimura, a bubbly 22 year old professional wrestler, committed suicide after receiving a stream of online abuse on Twitter and Instagram[5].

In the most extreme cases, dangerous online communities can radicalise and embolden disenfranchised individuals, leading some to commit horrendous crimes such as the New Zealand terrorist attack in 2019 where the perpetrator killed 51 people and injured 49, and broadcast-ed it live on the internet[4].

It cannot be overemphasised the importance of ensuring online safety in this digital age, both in terms of protecting individual mental health as well as achieving a tolerant and stable society. To do so we need to an effective way to detect offensive language in social media, one that would ensure civility of online discussions and prevent cyberbullying while allowing for opposing ideas to be expressed. Social Media

platforms like Facebook have traditionally argued that they are not content providers, but now countries like the UK are proposing to legislate to ensure that the platforms are putting systems in place to block harmful content on their sites[6].

1.2 Research Questions

This thesis aims to investigate the topic of Offensive Language Classification, taking advantage of recent breakthroughs in Natural Language Processing and Deep Learning techniques.

We present the results of our participation in a recent NLP research challenge focusing on detecting and classifying offensive language in Twitter by answering the following questions:

(A) Is the tweet offensive? (Offensive Language Detection)

(B) Is the offense targeted? (Classification of Offensive Type)

(C) If the offense is targeted, identify the target as one of the following - individual, group or others (Offensive Target Identification)

The questions are explored through the subtasks A, B and C, and discussed in detail in Chapter 5, 6 and 7 respectively.

1.3 Contribution

We participated in *"Task 12 - Offensive Language Classification Identification in Social Media"*[61] of SemEval 2020 Shared Task. Through improving data quality with the inclusion of additional data sources and employing various ensembling approaches, we yield a model that achieved a macro F1 score of 0,91344 in the English SubTask A, placing 11th place out of 82 teams. We published the result in a System paper[53] for the SemEval workshop.

1.4 Structure of the Thesis

This document is structured into eight chapters:

1. Chapter 2 introduces main concepts related to Natural Language Processing using Deep Learning and provides an overview of the major ideas leading up to the latest state of the art methods;

2. Chapter 3 introduces the three sub-tasks in Offensive Language Classification based on the OLID schema [59]. We analyse the datasets in detail and describe related work

3. Chapter 4 describes the experimental setup of our system;
4. Chapter 5 analyses the results of subtask A - offensive language detection;
5. Chapter 6 discusses the results of subtask B - offensive types categorisation, i.e. if the offense is targeted;
6. Chapter 7 describes the result of subtask C - offensive target identification, i.e. determining the target of the offense;
7. Chapter 8 concludes the thesis and provide recommendation for future work.

Chapter 2

Theoretical Background

The field of Natural Language Processing (NLP) has evolved over the years, moving from traditional statistically based machine learning approaches such as Latent Semantic Indexing[3], Latent Dirichlet Allocation[9], Brown Clustering[10], to the modern state-of-art deep learning methods like RNN[41], LSTM[25] and CNN[30]. Even within the deep-learning area, there has been significant developments in the last two years with the introduction of attention-based, Transformer[52]-type architectures like BERT[15] and GPT[44]. This chapter aims to summarise key concepts leading up to the development of the Transformer architecture and provide readers with a basic intuition of the BERT model[15].

2.1 Text Classification using Deep Learning

Offensive Language Classification is an example of a Text Classification Task in NLP, where a classifier automatically assigns labels to a group of texts based on its content. As text is sequential in nature, its meaning is derived from the individual words meaning as well as the order in which they are combined.

2.1.1 Word Representations

For deep learning models, the text input needs to be converted to numerical format. The simplest form is to *one-hot-encode* each word by giving it a unique integer value, then convert it into a binary vector of size N, the size of the vocabulary, where all values are zero except for the index of the integer, marked as 1.

Word embedding improves upon one-hot-encoding by creating a lower-dimensional representation of the words such that words with similar meaning will be grouped in the vector space. This is based on the idea of "distributional semantics", where a word's

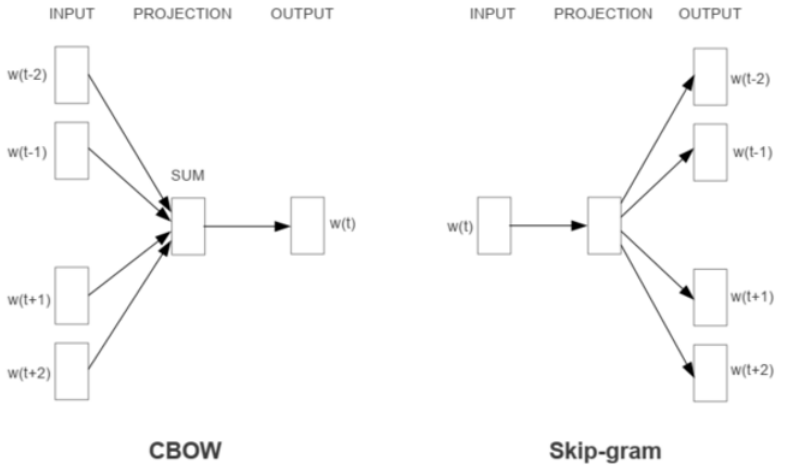

Figure 2.1: Word2Vec - The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word, from [36]

meaning is given by the words that frequently appear close-by [19, 22]. *Word2Vec* [36] and *GloVe* [42] are two popular algorithms for learning word embeddings inspired by neural network language model (NNLM)[8] where a feed-forward neural network with a linear layer and a non-linear hidden layer was used to learn the word vector representation and a statistical language model, i.e. a probability distribution over sequences of words.

Word2Vec[36] uses a concept of local context window where a target is surrounded by context words, and introduces a Continuous Bag-Of-Words (CBOW) algorithm to predict the current word based on the surrounding context words, and a Skip-gram algorithm to predict surrounding words given the current word as shown in Figure 2.1.

GloVe (Global vectors for word representation)[42] combines the concept of global matrix factorization and the local context window methods. Using the intuition that word meaning can be derived from its word co-occurrence probabilities, the model is trained to learn the weights of the word vectors by predicting global word co-occurrence counts.

These types of word embedding are often trained on a large corpus, and the representation are then saved as weights in an embedding matrix to be used as features for specific downstream tasks. These methods provides a good way of representing word meanings as well as their inter-relationships, but cannot represent words that have different meanings in different contexts. For instance, the word “bank” has a different meaning in the phrase “river bank” and “bank account”, but will have the same representation in this scenario. In the following section we will introduce new embedding representation that consider contextual information to embed each word in a latent space.

2.1.2 Sequence Modelling

The simplest form of representing a group of words like a sentence is with a *Bag-of-Words model*[22] where the model constructs a dictionary of unique words along with their word counts, disregarding grammar and word order. These become features that are fed into a linear classifier such as Naive Bayes[21], SVM[51] or Random Forest[23], to make a prediction.

Recurrent neural networks (RNN)[41] improves upon linear classifiers by taking into account the sequential nature of text. Figure 2.2 shows a single RNN cell, A, receiving an input X_t and producing an output h_t, and sending the output back to itself. At each time step t, the recurrent network A receives the input X_t as well as its own output from the previous time step h_{t-1}, allowing information to be passed from one time step to the next. This can be represented against the time axis as shown in Figure 2.2 (right) where RNN cell A is "*unrolled through time*". This chain-like nature allows RNN to model sequences such as text, where each time step represent a word in a certain position in a sentence. Figure 2.3 represents the case of text classification where a sequence of words, represented by the red boxes, are input into RNN (green boxes) at separate time steps and taking the last output, represented by the blue box, as the classification label.

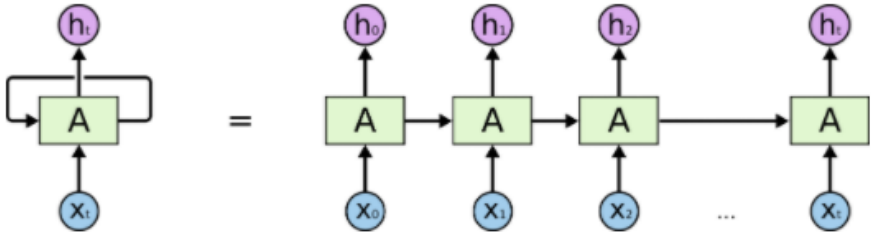

Figure 2.2: An unrolled recurrent neural network. From [40]

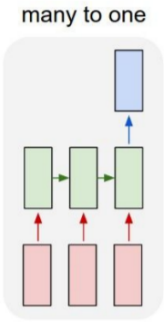

Figure 2.3: Sequence classification using RNN. Words are represented as input vectors in red, output vectors are in blue and green vectors hold the RNN's state. From [28]

An RNN cell learns its weights through back-propagation, and as the sequence length increases, the unrolled RNN becomes a very deep network. This causes the

gradient to either vanish or explode and as a result the network cannot learn long range information effectively.

The *Long Short-Term Memory* cell is a special kind of RNN that can detect long term dependencies in the data by using various gates to control and protect cell states. Figure 2.4 shows the content of an LSTM cell. The cell state is split into two vectors: $h_{(t)}$, the short-term state and $c_{(t)}$ the long-term state. As the long-term state $c_{(t-1)}$ traverses through the network, it goes through a *forget gate*, dropping some memories, then add some new memories that were selected by an *input gate* via the addition operation, resulting in the new cell state $c_{(t)}$. The long-term state is also copied and passed through a tanh function, and the result is filtered by the *output gate* to produce the short-term state, $h_{(t)}$ and the cell output at this time step, $y_{(t)}$

Now to understand at how the gates work. The current input vector $x_{(t)}$ and previous short-term state $h_{(t-1)}$ are fed to four different fully connected layers (FC) as shown at the bottom of Figure 2.4. The main layer analyses $x_{(t)}$ and $h_{(t-1)}$, then outputs $g_{(t)}$. The other three layers act as *gate controllers* that uses logistic activation function to output values between 0 and 1, where 0 represents a closed gate and 1 represents an open gate. The forget gate, $f_{(t)}$, controls which part of the long term memory should be erased; the input gate, $i_{(t)}$ controls which part of $g_{(t)}$ to add to the long-term state; the output gate, $o_{(t)}$ controls which part of the long-term state should be output to $h_{(t)}$ and $y_{(t)}$.

The ability of LSTM to preserve information and forget when it's no longer required makes it more successful than a basic RNN in capture long-term patterns.

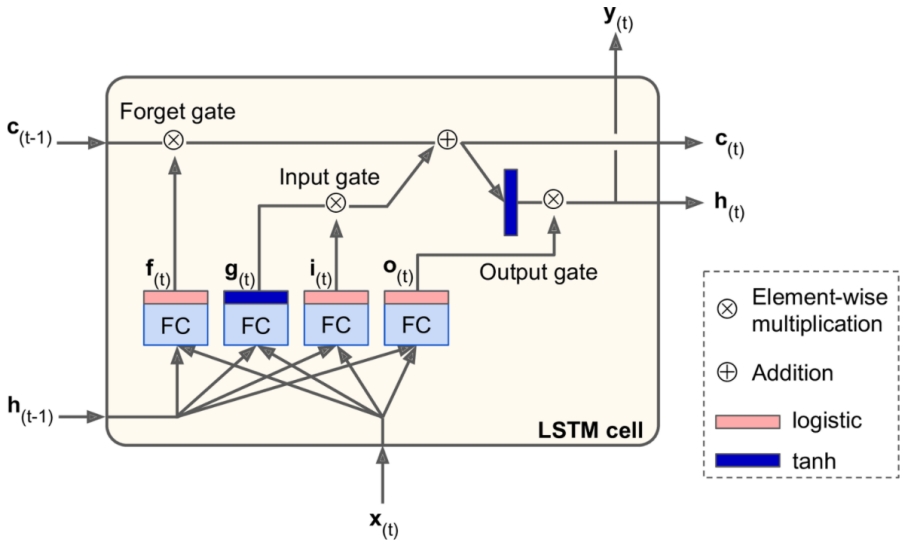

Figure 2.4: Architecture of a basic LSTM Cell, showing how the long-term memory $c_{(t)}$ and the short-term memory $h_{(t)}$ propagates through the cell controlled by the input gate, forget gate and output gate. $x_{(t)}$ and $y_{(t)}$ represent the input and output to the cell at time step t and fully connected layers are represented by FC. [18]

2.1.3 Attention

The *Attention* mechanism was introduced as a means of improving neural machine translation[2]. Neural machine translation generally uses an encoder-decoder architecture where a source sentence is encoded into a fixed-length vector from which a decoder generates a translation. The encoder-decoder is jointly trained to maximise the probability of a correct translation given a source sentence. However, when the source sentence is long, some information can be lost in the process of compressing the representation into a fixed length vector. This is also known as information bottleneck.

Attention solves the problem by allowing the model to focus on the relevant part of the sequence. For each step of the decoding process, there is a direct connection to every step of the encoder. An *attention score* is calculated to represent the similarity between the decoder state and each of the attention states, often using a dot product.

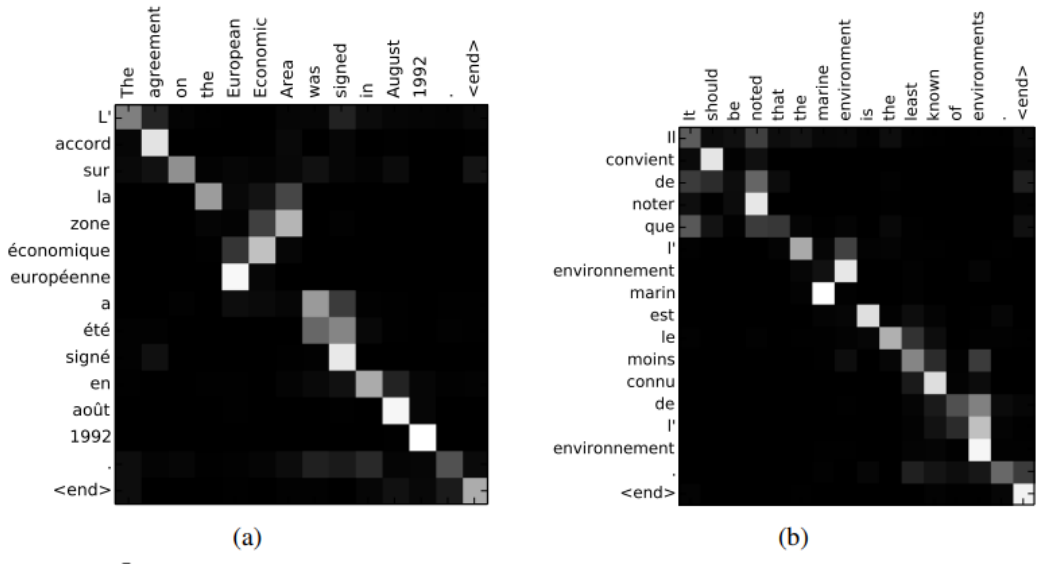

Figure 2.5: Attention patterns in generating French (left) to English (top) translation. Lighter value indicates more attention (higher weight) on that particular word. From [2]

2.1.4 Transformers

The Transformer[52] architecture reduces the amount of sequential computation with the use of attention mechanisms. Previously, Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM) and Gated Recurrent Units (GRU) were the dominant architectures for sequence modeling but their sequential nature precludes parallelization and do not work well for longer sequences. The transformer architecture utilizes only attention mechanisms to draw global dependencies between input and output, its capability for parallel operations resulted in a significant increase of efficiency.

The transformer architecture consists of an encoder and a decoder component, shown as the left and right halves of Figure 2.6. Both the encoder and the decoder consist of six stacked multi-head self-attention layers and point-wise, fully connected layers.

The self-attention layers in the encoder uses keys, values and queries from the previous layer or from the input embedding. Each position in the encoder represents a single word and can attend to all positions in the previous layer, such that the influence of the other words in the sentence on the current word is taken into account when the word is encoded. Using "Scaled Dot-Product Attention", the calculation is done for all words simultaneously using matrices $Q \in \mathbb{R}^{N_s \times d_k}$ (Query) , $K \in \mathbb{R}^{N_t \times d_k}$ (Key) and $V \in \mathbb{R}^{N_t \times d_v}$ (Value) by computing the dot products of the query with all keys, where N_t denotes the target vocabulary size N_s the source vocabulary size and d_k the latent space dimension for keys and queries and d_v for values. We divide each by $\sqrt{d_k}$, square root of the dimension of the keys and queries, and apply a softmax function to get the weights on the values as shown by the equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Multi-head attention is created by projecting queries, keys and values h times with different, learnt linear projections to d_k, d_k and d_v dimensions, respectively. The attention function is then preformed in parallel on each of the projected versions, yielding d_v-dimensional output values. Finally these values are concatenated and again projected, resulting in the final values. (Figure 2.7). The use of multi-head attention allows the model to jointly extract information from different representation subspaces at different positions.

For the decoder layer, there is a multi-head self-attention sub-layer, a fully connected feed forward network as well as an additional encoder-decoder attention sub-layer. The self-attention layer allows each position in the decoder to attend to all positions in the decoder up to and including that position, and uses masking to prevent the position from attending to subsequent positions. The encoder-decoder layer uses queries from the previous decoder layer, and the memory keys and values from the output of the encoder, to allow every position in the decoder to attend over all positions of the input sentence.

Input and output words are converted into vectors of dimension d_{model} using learnt embeddings. The decoder output is converted with learned linear transformation and softmax function, to predict the next-token probability.

In order to encapsulate the relative position of the tokens in the sequence, "positional encoding" is added to the input embeddings at the bottoms of the encoder and decode stacks. The positional encodings have the same dimension, d_{model}, as the embeddings, so they can be summed.

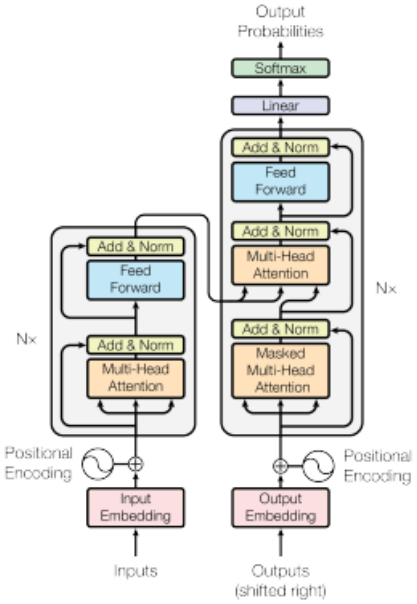

Figure 2.6: The Transformer Architecture - with stacked multi-head self attention (orange) and fully connected layers (blue) in both encoder (left) and decoder (right). Input to the model is a sequence of words and the output is a sequence of word probabilities. From [52]

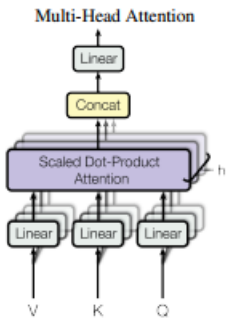

Figure 2.7: Multi-Head Attention consisting of h attention layers running in parallel. These layers are then concatenated and passed through a linear layer. From [52]

The Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers as its operations are highly parallelizable. It also achieved a new state of the art results for translation tasks and became the basis for a series of subsequent milestones in NLP.

2.1.5 Contextualised Embeddings

BERT (Bidirectional Encoder Representation from Transformer)[15] is a cumulation of several ideas in NLP coming together to create a major breakthrough.

BERT uses a multi-layer bidirectional Transformer encoder architecture, based on implementation described in Vaswani et al. [52]. To understand the concepts behind BERT, it is useful to see the some key progressions in NLP leading up to its inception.

Previous work has shown "Language Model Pre-training" to be an effective technique for learning word embeddings in NLP tasks[13, 26, 43, 44], whereby training models to predict the next word in a sentence allow the models to acquire familiarity with the language without the need of labelled data. These word representations benefit from having models trained on large corpus of data such as Wikipedia and published books.

One such example is ELMo[43], which uses a bi-directional LSTM to extract *context-sensitive* word representations from a left-to-right and a right-to-left language model. Contextual word embeddings are created by concatenating these representations, and are used as input features in task-specific down-stream architectures.

ULMFit[26] extends the idea of Language Model Pre-training by introducing a Language Model fine-tuning method for classification tasks that mimics Computer Vision-style transfer learning. Rather than just learning and extracting contextualized embeddings, the ULMFit model utilizes the same architecture for the unsupervised pre-training and the downstream fine-tuning. Firstly the model is trained on general-domain corpus to capture features of the language in different layers, then the target tasks is used to fine-tune the language model to obtain task-specific features. Finally the target task classifier is fine-tuned through gradual layer unfreezing, allowing the model to preserve low-level representations such as the embeddings, while adopting high level ones. The ULMFit uses a regular bi-directional LSTM architecture, with no attention or residue connections, and various tuned dropout hyperparameters. (Figure 2.8)

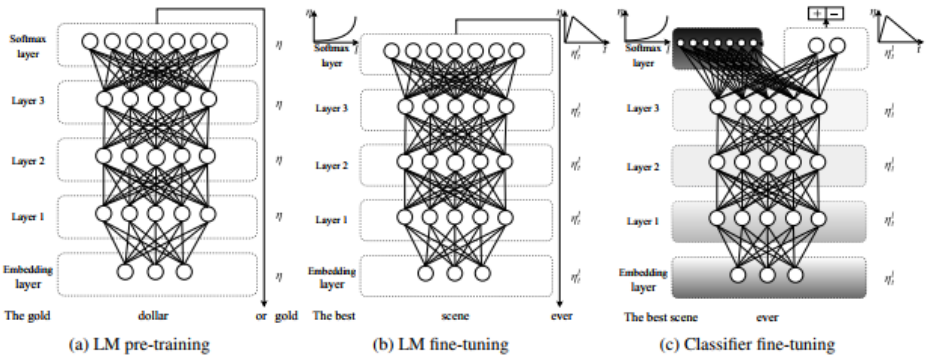

Figure 2.8: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('Discr') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, 'Discr', and STLR top reserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen) From [26]

The OpenAI Transformer, GPT[44] continues with the idea of pre-training a Language Model on large corpus of unlabelled text, followed by discriminative fine-tuning on target task. Unlike the bi-LSTM-based ULMFit, OpenAI GPT utilizes a 12-layer decoder-only transformer with masked self-attention heads. It also supports down-stream tasks beyond text classification, such as question answering, semantic similarity assessment and entailment determination. Only the decoder part of the transformer architecture is used, as it fits the purpose of language modeling (predicting the next

word). During the unsupervised pre-training, the BooksCorpus dataset[63], containing 7,000 unpublished books, is used to train a forward-only language model. At the fine-tuning stage, the model is tuned to optimise both the target task as well as the initial language model. This has been shown to improve the generalization of the supervised model as well as accelerate convergence. This type of fine-tuning based approach allows for task-agnostic architecture. The attention mechanisms of the transformer is able to better capture the language patterns and support transfer learning compared to LSTM, and as a result OpenAI achieved state of art results on 9 of 12 NLP datasets at the time.

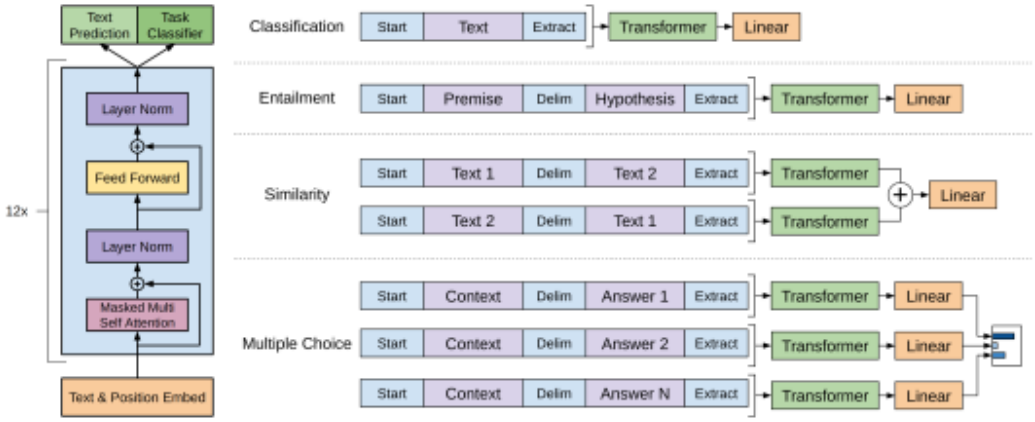

Figure 2.9: (left) Transformer architecture and training objectives used in GPT. (right) Input transformations for fine-tuning on different tasks. All structured inputs are converted into token sequences to be processed by the pre-trained model, followed by a linear+softmax layer. From [44]

2.1.5.1 BERT

Similar to ULMFit[26] and OpenAI GPT[44], BERT[15] also utilizes the *pre-training* and *fine-tuning* approach using language models. BERT has a unified architecture across different tasks, with minimal difference between pre-trained architecture and the final downstream architecture.

Whereas OpenAI GPT uses a multi-layer forward-only Transformer decoder architecture, BERT is based on a multi-layer bi-directional Transformer encoder architecture. BERT is pre-trained using two unsupervised tasks - *Masked Language Model (MLM)* and *Next Sentence Prediction (NSP)*.

In a multi-layered environment like the Transformer, if the language model is trained from both left-to-right and from the right-to-left, the word will inevitably "seeing itself" in other layers. This presents a challenge in achieving bi-directionality in Transformers. BERT overcame this restriction by using *Masked Language Model(MLM)* - randomly mask 15% of the input text and train the language model to predict the masked words instead of the next word in a sequence. This allows BERT to be a more

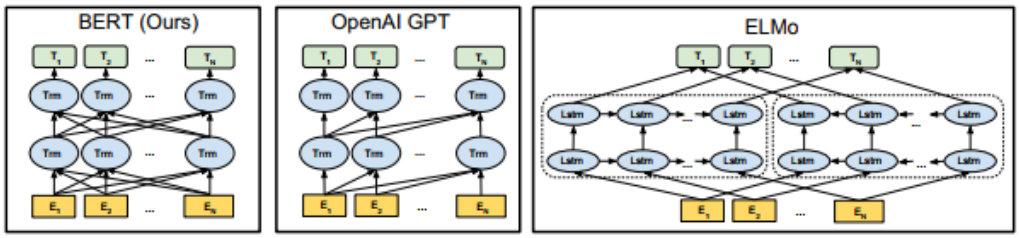

Figure 2.10: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-toleft LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach. From [15]

powerful, deep bi-directional model than a shadow bi-directional model produced by concatenating a left-to-right and a right-to-left model such as ELMo.

To cater for downstream tasks that requires an understanding of relationships between sentences such as Question Answering (QA) and Natural Language Inference (NLI), BERT introduces a *Next Sentence Prediction (NSP)* task for the pre-training stage. Here, sentences A and B are chosen from the corpus where 50% of the time B is the next sentence that follows A, and 50% of the time B is a random sentence from the corpus. The model is then trained to identify if sentence B follows A, in doing so, it learns to understand the relationship between two sentences.

BERT is pre-trained using the BooksCorpus[63] containing 800M words and the English Wikipedia, containing 2,500M words. The size of the corpus allows the pre-trained model to develop deeper understanding of the underlying language patterns. Two BERT pre-trained models are available for download: $BERT_{BASE}$ and $BERT_{LARGE}$. $BERT_{BASE}$ consists of 12 layers, hidden size of 768 and 12 self-attention heads, totalling 110M parameters and $BERT_{LARGE}$ has 24 layers, 1024 hidden units, 16 self-attention heads and a total of 340M parameters. The pre-training process take a significant amount of time, but do not need to be repeated as the downstream target task models are initialized using all of the pre-trained model weights.

The downstream model has essentially the same architecture as the pre-trained model, but with an additional task-specific output layer. After plugging task-specific inputs and outputs into the BERT model, all the parameters are fine-tuned end-to-end. For tasks involving sentence pairs, such as question-answering, BERT´s self attention mechanism applies bi-directional cross attention between the two sentences to learn their dependencies.

Figure 2.11 shows the structure of BERT´s input representation. The input representation can represent both a single sentence and a pair of sentences in one token sequence, based on the downstream task. The first token of every sequence is a special

classification token ([CLS]), whose final hidden state represents the sentence aggregation results for classification tasks. For tasks requiring sentence pairs, the pairs are packed together into a single sequence, separated with a special token ([SEP]). A segment embedding is added to the token embedding to differentiate sentence A from sentence B. Finally the position embeddings are added to the input representation.

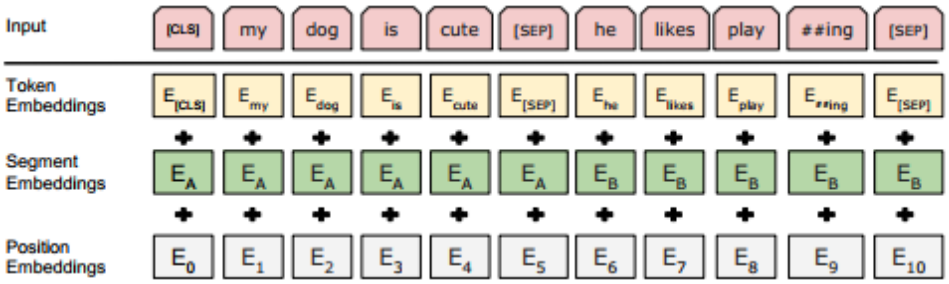

Figure 2.11: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. From [15]

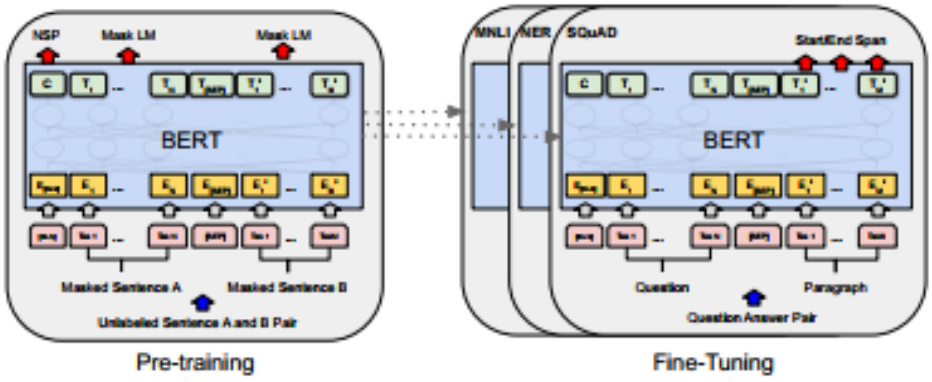

Figure 2.12: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers). From [15]

*

Chapter 3

Offensive Language Classification Tasks & Data

The abusive and offensive language classification task has been studied from different perspectives, ranging from aggression identification [29, 37], bullying detection [57], hate speech identification [14], toxic comment classification [16], and offensive language classification [55, 60].

In a recent comparative study of different subtasks related to abusive language detection, Waseem et al. [54] suggested the subtasks can be synthesised with a typology that captures their central similarities and differences. They proposed a two-level typology that considers whether (i) the abuse is directed to a specific target, and (ii) the degree to which it is explicit. Directed abuse, where someone is mentioned by name, tagged by a username or referenced by a pronoun are examples of cyberbullying, while abusive expressions towards generalised groups such as racial categories can be considered as hate speech. By drawing a distinction between explicit and implicit abuse, they highlighted the difficulties in detecting the latter, which can often be obscured by the use of ambiguous terms, sarcasm and lack of profanity and hateful terms.

Zampieri et al. [59] extended this idea and compiled a Offensive Language Identification Dataset (OLID) that follows a three-level hierarchical schema for offensive language classification which considers:

(A) whether a tweet is offensive or not (Offensive Language Detection)

(B) whether the offense is targeted (Categorization of Offensive Language Types)

(C) whether it is targeted towards individuals, groups or others. (Offensive Language Target Identification)

They proposed that the fine-grained approach would help to consolidate related tasks in this field, and including the target of the abuse would assist future studies of hate speech with respect to a specific target.

The OLID [59] dataset became the official dataset for the OffensEval 2019 Shared Task [60] where each level correspond to a specific sub-task. OffensEval 2020 continued with the same hierarchical annotation schema but with much larger, semi-supervised datasets.

In this chapter we analyze the official dataset for OffensEval2020 English sub-task A [61], and discuss additional datasets and resources that we used for this task. We also describe the official dataset for OffensEval 2019 sub-task B and sub-task C [60] and look at related work for all the subtasks.

3.1 Level A - Offensive Language Detection

3.1.1 Task Description and Dataset

According to OLID [59] annotation guidelines, level A discriminates between the following types of tweets:

- Not Offensive (NOT): Posts that do not contain offense or profanity
- Offensive (OFF): Posts containing any form of non-acceptable language (profanity) or a targeted offense, which can be veiled or direct. This includes insults, threats, and posts containing profane language or swear words.

3.1.1.1 Semi-Supervised Dataset for Offensive Language Identification (SOLID)

The official OffensEval 2020 English Training Dataset (**SOLID**)[45] contains over nine million tweets and was labelled in a semi-supervised manner using model built from an ensemble of PMI [50], LSTM [24], FastText [27] and BERT [15].

It followed the annotation guideline as OLID, but instead of the binary label, two numerical scores are provide for each tweet – μ and σ. μ represents the average of the confidences predicted by the models for belonging to the positive class (OFF). σ is the confidences' standard deviation. Table 3.1 shows a sample of the dataset content.

id	text	average (μ)	std(σ)
1159533701283350000	First time I heard his name in camp, he seems to be the forgotten guyChristian Covington being disruptive so far today	0.195773	0.187379
1159533703522990000	When I go to drink with Tsubaki he would always fall asleep first. His sleeping face looks really innocent and not like him at all. Fufu♪	0.262401	0.145998
1159533703758060000	@USER His ass need to stay up	0.833391	0.140628

Table 3.1: SOLID dataset sample entries

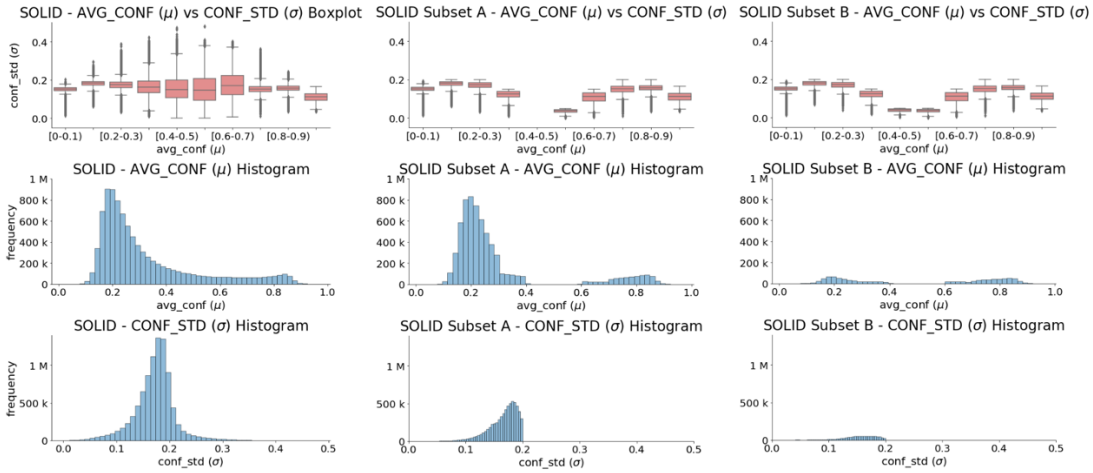

Figure 3.1: SOLID Data Distributions - Original and Subsets

In order to find a suitable decision boundary for binary classification, we binned μ and σ and analysed their respective distribution using histograms for the confidence levels (Figure 3.1-middle) and standard deviation (Figure 3.1-bottom), as well as the distribution of the standard deviation over the confidence scores (Figure 3.1-top).

The original dataset is shown as the left-most column on Figure 3.1. The middle graph, *SOLID - AVG_CONF (μ) Histogram*, shows the dataset has significantly more lower values of average confidence (μ) such that there will be more negative examples (NON) than positive (OFF) if the threshold is taken around a mid-range value of 0.5. The boxplot on the top left column shows that mid-range values of μ have a higher variance in the σ values.

We inspected the tweets with mid-range values of μ ($0.4 < \mu < 0.6$) and high standard deviation σ (> 0.3) and found various examples of mis-classification (assuming that the classification threshold was set at $\mu = 0.5$).

Table 3.2 shows examples of offensive texts with $\mu < 0.5$. Surprisingly many of these tweets contain profanity but were not correctly identified. This suggested the semi-supervised training algorithm was not very sensitive to the profanity words.

On the other hand, Table 3.3 highlights that self deprecating comments were often mislabelled as being offensive and general negative comments on society or negative sentiments were also incorrectly marked as offensive.

As tweets with mid-range μ and / or high σ appear to be inconsistent and misleading, we decided to remove these uncertain regions from the training dataset instead of altering the threshold. We created two subsets by under-sampling the mid-range values of μ ($0.4 < \mu < 0.6$) to see if a more selective sampling would improve the results, one containing about 6M samples by just under-sampling the uncertain region (SOLID A) and another with only 1M samples (SOLID B).

text	average (μ)	std (σ)
First of all bitch Let me take Lay to Walmart before I push her down	0.400922	0.339073
Not in da mood 4 these hoes	0.401108	0.321429
my favorite part of coming to the gym alone is that all the persian fuckboys come out to play	0.420394	0.321399
But America is the shittiest place on earth right?? ??	0.423781	0.319078
I do love all the slutty costumes though. Even though half of you are sluts anyways.	0.424623	0.314600
religion is so fuckingggff bulshittt	0.426189	0.300384
@USER Tell it to the IMBECILE in the White House! His name is #Trump	0.443468	0.314032
@USER You are Corrupted! You should retire right now and step down!	0.480829	0.313158
Sooo many fresh when we was in school niggas bums now....tf happen to u!	0.485208	0.318131
Thick master Ken is certainly one of the nastiest sissy boys of all time	0.436352	0.303370
So Lion King was absolute dogshit	0.442239	0.315102
@USER attention WHORE be gone.	0.493955	0.320506
The Chinese Satanist	0.490647	0.327647
@USER only If you are weak and insecure	0.489890	0.308845
Losing to Retards	0.496200	0.303528

Table 3.2: Example of Mis-classification - offensive texts with $\mu < 0.5$ in SOLID

text	average (μ)	std (σ)
Today sucked. Where's the tequila?	0.501158	0.321304
caring about something you know is pointless	0.503435	0.311307
fear and greed	0.504076	0.305328
This heat is truly obnoxious	0.504384	0.303848
My body is garbage and I would love to rent a new one pls.	0.523693	0.325015
Let me start on this book to kill some time	0.536988	0.323009
You know for a bimbo I sure do read a lot...	0.538729	0.343389
in a world that needs a lot of help i feel so useless ...	0.545438	0.302742
Ate veggies today and I'm not thinner...damn it!	0.547136	0.314980
Animals is one hell of an album	0.547262	0.312807
The world is cruel #bot	0.547752	0.317849
@USER It's coming. Be patient	0.548401	0.325815
Ignorance is a bliss	0.548738	0.300488
the graveyard 2 was insane	0.549510	0.331272
@USER i will, thank you. this week was crap.	0.550122	0.319092
Still missing you like crazy tho'	0.557427	0.338659
I am just too sad right now and i feel worthless...	0.557784	0.347495
I've really, really been the best of fools, I did what I could	0.561318	0.300886
Why do fools fall in love?	0.575715	0.330991

Table 3.3: Example of of Mis-classification - non-offensive texts with $\mu > 0.5$ in SOLID

SOLID A was created by removing majority of records with mid-range μ as well as removing records with $\sigma > 0.2$. The distribution is shown in middle column of Figure 3.1. The resulting dataset contains 6,464,288 records, roughly 72% of the original dataset volume.

SOLID B was created with the intention of balancing dataset by sampling equal number records from both high and low μ ranges. Records with $\sigma > 0.2$ were removed and only a small number of tweets with mid-range μ were kept. The distribution is shown in the right column of Figure 3.1. The resulting dataset contains 1,030,000 records, roughly 11% of the original dataset volume.

3.1.2 Data Augmentation

We opted to augment the dataset as the semi-supervised nature of SOLID lead to more ambiguity in its usage and as well as potentially being less reliable in the results. The dataset we chose to include are described in the following section. Table 5.2 shows the target distributions of the datasets we used.

Class	SOLID	OLID Train	Kaggle	Profanity	SOLID A	SOLID B
NON	7,628,650 (84%)	8,840 (67%)	1,660,540 (92%)	147,509 (80%)	5,636,935 (87%)	515,000 (50%)
OFF	1,446,768 (16%)	4,400 (33%)	144,334 (8%)	36,845 (20%)	827,353 (13%)	515,000 (50%)
TOTAL	9,075,418 (100%)	13,240 (100%)	1,804,874 (100%)	184,354 (100%)	6,464,288 (100%)	1,030,000 (100%)

Table 3.4: Target Distribution of the datasets.

3.1.2.1 Offensive Language Identification Dataset (OLID)

The Offensive Language Identification Dataset (OLID)[1] was created for the OffensEval 2019 shared task. The training set contains 14,100 manually annotated tweets, where a tweet was labelled as offensive (OFF) if it contains any form of profanity or targeted offense, either veiled or direct, and non-offensive (NON) otherwise. The ratio of OFF to NON is roughly 1 to 2. The quality of this dataset is better and more reliable than SOLID, however it is almost 650 times smaller.

id	tweet	subtask_a
86426	@USER She should ask a few native Americans what their take on this is.	OFF
18485	@USER I am truly sorry that you are having a rough day. I hope it gets better for you. I am doing fantastic.	NOT
24276	@USER He is truly dumb as shit.	OFF
62688	@USER Someone should've Taken this piece of shit to a volcano.	OFF
95295	...and what plan might that be? The discredited Chequers Plan or crash out without a plan? URL	NOT

Table 3.5: OLID dataset sample entries

[1] https://scholar.harvard.edu/malmasi/olid

3.1.2.2 Jigsaw Unintended Bias in Toxicity Classification Dataset (Kaggle)

The Kaggle 2019 Toxicity Classification Dataset (Kaggle)[2] dataset contains over 1.8 million public comments from online news discussions. This dataset was created with the aim of reducing unintended bias in toxicity classification as a result of identity mentions. The data has been labelled with identity mentions, such as Muslim, Gay or Black, and a toxicity score (TARGET) that represents the faction of human annotators who believe the post is toxic. We decided to include this as it is a large dataset where each comment has been reviewed by up to 10 annotators, and its content could prove useful in reducing false positive errors. For this task we will just use the comment_text and target columns. As the dataset contains comments rather than tweets, the length of text could be significantly longer.

id	target	comment_text
5672903	0.000000	Just look at the comments here in the SA, every single little snowflake (D)onkey is against the wall. Obviously you do know that the money will be spent in the US to build the wall which means lots of manufacturing and construction jobs for US workers.
6046443	0.000000	I'm guessing it will bounce or Trump will simply reimburse him behind the scene.
6156401	0.868852	Pelosi needs to go she is such an idiot the craziest person that I have had the non privilege of trying to comprehend anything that comes out of that disgusting mouth of hers CLEAN THE SWAMP she is nuts
5158315	0.000000	What do you expect from a newspaper that has less than 100 real subscribers? LOL

Table 3.6: Kaggle dataset samples, showing *target* and *comment_text* columns

3.1.3 Additional Resources

3.1.3.1 Profanity Check (Profanity)

A simple text search of the word "fuck" in SOLID returned 268,845 matches, roughly 3% of all tweets. Out of these, 2.3% were mis-classified as non-offensive. Figure 3.2 compares the distribution of the confidence scores μ and σ for profanity tweets containing the word "fuck" against those in the whole dataset. The box-plot on top left show that tweets with μ between $0.3 - 0.7$ have a larger σ, indicating more uncertainty in the classification. The SOLID-AVG_CONF(μ) histogram in the middle left also shows some tweets with the word "f**k" had $\mu < 0.5$ (Figure 3.2).

This suggests a need to increase our model's sensitivity to profanity words. Rather than using a dictionary based approach [20], we decided to use a Python library, *Profanity-check*[3], that checks for profanity and offensive language in text. It was built using a SVN model trained on 184,354 records from a twitter dataset[4] and a Wikipedia dataset [62]

[2] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification
[3] https://pypi.org/project/profanity-check/
[4] https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data

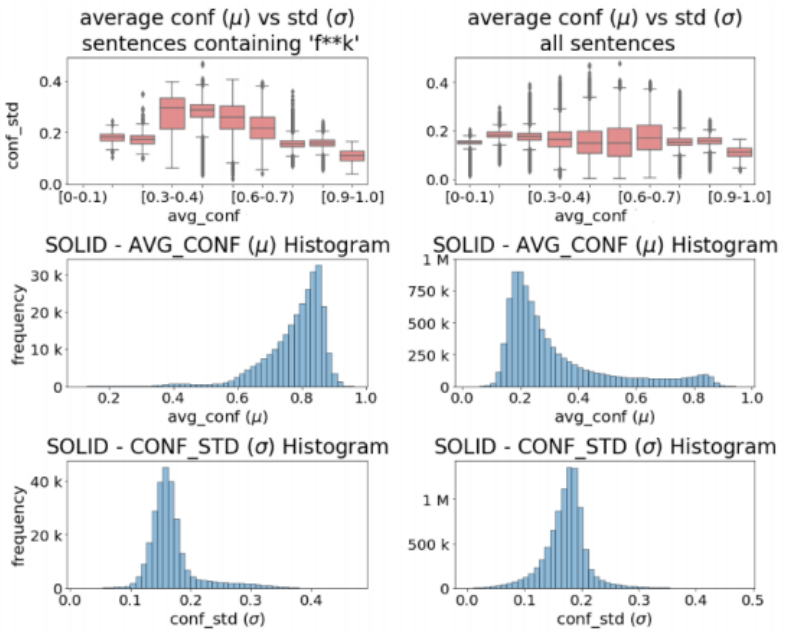

Figure 3.2: Distribution of f**k words in SOLID

3.1.4 Using the Datasets

Having sourced additional dataset for this task, we will create separate models using data from each dataset and compare their performances in Chapter 5.

3.1.5 Related Work

OffensEval 2019 sub-task A used the OLID dataset and 104 teams participated in the task. Seven out of the top ten teams used BERT, with various pre-processing techniques and different parameters.

The top team, *NULI*[31] experimented with different models including Linear, LSTM and fine-tuning pre-trained BERT on the OLID dataset. They used pre-processing techniques such as emoji substitution, hashtag segmentation, replacing URLS and converting text to lowercase. Their final model was BERT, as it outperformed other models during development.

The second team, *vradivchev_anikolov*[38] trained a large number of models including Naive Bayes, SVM, CNN, MLP, RNN and BERT, then combined the best of them in ensembles. Their pre-processing step included removing most punctuation except for "@" and "#", hashtag segmentation and stop words removal. Their best performing model was also BERT, as the other models overfitted in Dev.

The top non-BERT model, *MIDAS*[33], used an ensemble of Convolutional Neural Network, Bidirectional LSTM with attention, and Bidirectional LSTM + Bidirectional GRU and is ranked 5th.

3.2 Level B - Automatic Categorization of Offense Types

3.2.1 Task Description and Dataset

Using OLID [59] annotation guidelines, level B differentiates between the types of offense by detecting if the offense is targeted:

- Targeted Insult (TIN): Posts containing insult/threat to an individual, a group, or others;
- Untargeted (UNT): Posts containing non-targeted profanity and swearing. Posts with general profanity are not targeted, but they contain non-acceptable language.

We will use the Offensive Language Identification Dataset (OLID)[5], which was also the official dataset for OffensEval 2019 subtask B. Table 3.8 shows how different tweets are labelled as TIN or UNT in column *subtask_b*. We note that some scenarios can be ambiguous, or even incorrectly labelled. For instance the last sample in the table contains the text "I just threw up in my mouth." appears to be untargeted and was labelled as such. But the hashtag #LockHerUp suggests otherwise. This highlights the complexity of this subtask and perhaps the need for more detailed annotation guidelines and cross checking.

The distribution of TIN and UNT are fairly unbalanced, with 3,876 targeted offenses and 524 untargeted offenses. (Table 3.7)

A	B	C	Training	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Table 3.7: Distributions of label combinations in OLID

3.2.2 Related Work

This is an existing task and results and papers have been published in OffensEval 2019 task report[60]. 76 teams participated in the sub-task B. Unlike sub-task A, offensive language detection, where majority of the top models are BERT-based, the offense type classification task drew out a wider variety of techniques. In particular five out of the top ten teams used an ensemble model.

The best team, *jhan014*[20], devised a probabilistic model to calculate the offensiveness of a sentence based on the offensiveness of individual words in the sentence and

[5] https://scholar.harvard.edu/malmasi/olid

tweet	subtask_a	subtask_b	subtask_c
@USER @USER @USER Bad people don't obey the law. There are many more guns in circulation in North America than there are people. Gun control is liberal mind masturbation.	OFF	TIN	GRP
@USER Comparing a date to rape is disgusting. I hope your family never has to deal with rape. You are a disgusting POS.	OFF	TIN	IND
@USER He is repulsive.	OFF	TIN	IND
@USER @USER @USER Fake conservatives and their fake outrage over the budget betrayal URL URL	OFF	TIN	GRP
@USER CNN = Fake News	OFF	TIN	OTH
@USER @USER These liberals have no problem lying.	OFF	TIN	OTH
@USER this is fucking hilarious	OFF	UNT	NaN
I just threw up in my mouth. #LockHerUp #MAGA URL	OFF	UNT	NaN

Table 3.8: OLID samples for Offense Type Categorisation and Target Identification

intensified by other words in the sentence that are syntactically related to the offensive words. The inclusion of syntactic information in their model assisted in the detection of targeted offense because usually targeted offensive language have different sentence structure with untargeted ones.

The second team, *Amobee*[46] designed a new type of convolutional neural network called "Multiple Choice CNN" (MC-CNN) and used that over a custom developed embedding. The idea was to replace quantitative questions such as "how mad is the speaker?", where the result is believed to be represented by the activation of the corresponding filter, with multiple choice questions such as "what is the speaker - happy/sad/other?", where the number of choices denoted by the number of filters and the sum of filter activations for each group is forced to be equal to 1. The approach appear to result in less over-fitting.

The best team from sub-task A, *NULI*[31], again fine-tuned a BERT pre-trained model and was ranked 4th, showing that BERT worked well for this task too.

3.3 Level C - Offense Target Identification

3.3.1 Task Description and Dataset

After the data has been identified as Targeted (TIN) in level B, level C of OLID [59] annotation schema is about detecting the target of the offense by categorising the target as the following:

- Individual (IND): Posts targeting an individual. This can be a famous person, a named individual or an unnamed participant in the conversation. Insults and threats targeted at individuals are often defined as cyberbullying.
- Group (GRP): Posts targeting a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or other common characteristic. Many of the insults and threats targeted at a group correspond to what is commonly understood as hate speech.

- Other (OTH): The target of these offensive posts does not belong to any of the previous two categories (e.g., an organisation, a situation, an event , or an issue).

Table 3.7 shows that the Target distribution is again unbalanced, with significantly more IND labels than GRP and OTH. Table 3.8 shows samples of the categorisation in the column *subtask_c*. While posts targeted towards individuals are easier to identify, there can be some confusion between GRP and OTH labels. In the table, the post "USER USER These liberals have no problem lying." is classified as OTH, but it can be argued that "liberals" infer a political affiliation and should be classified as GRP.

3.3.2 Related Work

OffensEval 2019 task report[60] also included team results on this sub-task. Similar to sub-task B, ensembles were quite popular and were used by five of the top ten teams.

The best team, *vradivchev_anikolov*[38] experimented with a large number of models such as SVM, Naive Bayes, RNN, MLP and BERT and combined the best models in a soft voting classifier. Instead of using a default classification threshold of 0.5, they derived optimal thresholds for each class using cross-validation. BERT was their best model, as the other models, including ensemble, overfitted to the training set.

The second team, *NLPR@SRPOL*[48] combined in an ensemble several models (LSTM, Transformer, OpenAI's GPT, Random forest, SVM) with various embeddings (custom, ELMo, fastText, Universal Encoder) together with additional linguistic features(number of blacklisted words, special characters, etc.). Their model was trained on publicly available datasets as well as their custom datasets annotated by linguists.

Chapter 4

Experimental Setup

BERT[15] has revolutionised the field of NLP since its inception in late 2018, producing state-of-the art results across many NLP tasks, including text classification. As discussed in the previous section, BERT-based models performed well across all three OffensEval 2019 subtasks, achieving 1st place in subtask A and C and 4th place in subtask B. The alternative models that achieved comparable results in training required far more task specific and feature engineering yet often over-fit during test.

As a result, we chose to use pre-trained BERT models for all three subtasks, and fine-tune these models using downstream datasets for each task.

There are many benefits of using a pre-trained BERT model. As the model was trained on a large corpus of data, it can produce fairly good results even if the dataset used for fine-tuning is relatively small in size as the model architecture remain the same and only the weights are adjusted.

HuggingFace [56] provides a well-developed API and includes downloadable pre-trained models. We will use BERT-base-uncased model, which contains 110M parameters.

4.1 Text Pre-processing

For Twitter datasets OLID (§ 3.1.2.1) and SOLID (§ 3.1.1.1) the text were pre-processed using the following steps:

- replace emojis with text descriptions, using Python library, emoji[1]
- expand hashtags and contractions
- replace Ampersand (&) with "and"

[1] https://pypi.org/project/emoji/

- replace "URL"with "http"
- keep a maximum of three consecutive occurrences of "@USER"
- replace tabs and line feed characters with a single blank space
- remove all symbols except for the following [-?.,!@#].
- convert text to lowercase

For the Kaggle dataset (§ 3.1.2.2), we simply expanded the contractions and converted text to lowercase.

4.2 BERT & HuggingFace

We implemented BERT using the PyTorch-Transformers library (https://pytorch.org/hub/huggingface_pytorch-transformers/) from HuggingFace. [56].

The following steps prepare the data into a suitable input format:

1. Add “[CLS]” at the beginning and “[SEP]” at end of each sentence
2. Tokenize text using the pre-trained BERTTokenizer (“BERT-base-uncased”)
3. Pad each sequence to a specified maximum length (128 for twitter datasets)
4. Create attention masks which contains 1s for each token and 0s for padding
5. Split the data into training and development sets and convert the data into Torch Tensors
6. Create training and development data-loaders

We used these steps to train the model:

1. Load a pre-trained “BERT-base-uncased” BERTForSequenceClassification model from PyTorch-Transformer library. This model extends the base BERT model by the inclusion of a final linear layer corresponding the number of output classes.
2. Use AdamW optimizer, with learning rate of $lr = 2e^{-5}$
3. Use Warmup Linear Schedule where the number of warm up steps is between 5-10% of total number of steps
4. Train the model between 2-3 epochs based on the performance on the development set
5. Combine training and validation set and use the same parameters to retrain model

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative (tn)	False Positive (fp)
	Positive	False Negative (fn)	True Positive (tp)

4.3 Evaluation Metrics

A confusion matrix is often used to describe the performance of a classification model and consists of the number of observations belonging to each of the following:

True Positives (tp) - correctly predicted positive values
True Negatives (tn) - correctly predicted negative values
False Positives (fp) - when the actual class is no but the predicted class is yes
False Positives (fp) - when the actual class is yes but the predicted class is no

The official metric for this task **macro-F1**, giving equal importance to **precision** and **recall**, as well as equal weighting on the minor and major classes. We will also use **macro-precision**, **macro-recall** as well as **accuracy** as intermediate steps in our evaluation. **Precision**, **recall** and **accuracy** are calculated as follows:

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn} \qquad acc. = \frac{tp + tn}{tp + fp + tn + fn}$$

F1 score is the weighted average of **precision** and **recall**. It is often more useful than **accuracy** especially when the class distribution is unbalanced.

$$F1 = \frac{precision * recall}{precision + recall}$$

Macro-averaged F1-score, or the **macro-F1**, is computed as an arithmetic mean of individual per-class F1-scores:

$$macroF1 = \frac{1}{C} \sum_{i=1}^{C} F1_i$$

where C is the total number of classes.

Macro-F1 is a good metric to use for our tasks as due to the imbalanced data the performance of the model on minority class will have a significant impact on the results.

Similarly, **macro-precision** and **macro-recall** are the arithmetic mean of their individual per-class scores:

$$macro - precision = \frac{1}{C} \sum_{i=1}^{C} precision_i$$

$$macro - recall = \frac{1}{C} \sum_{i=1}^{C} recall_i$$

where C is the total number of classes.

Chapter 5

Offensive Language Detection (Level A)

5.1 Training & Dev Split

We created train/dev splits with the sizes specified in Table 5.1 to find the best hyper-parameter for training each individual model. We then used OLID test set for evaluate and compare the results of each model.

Model	Training Set Size	Validation Set Size
OLID	11,916	1,324
SOLID	9,075,365	50,000
SOLID subsample A	6,444,288	20,000
SOLID subsample B	1,010,000	20,000
Kaggle	1,624,387	180,487
Profanity Check	N/A	N/A

Table 5.1: Training and Validation Set Sizes, for model tuning.

Class	OLID Test
NON	620 (72%)
OFF	240 (28%)
TOTAL	860 (100%)

Table 5.2: Target Distribution of Dev dataset, for model selection.

Model	macro-F1	macro-P	macro-R	acc.
SOLID	0.80967	0.84332	0.78925	**0.85814**
SOLID A	0.78802	0.77548	0.81384	0.80833
SOLID B	0.80237	0.79466	0.81223	0.83605
OLID	**0.81122**	0.80278	**0.82218**	0.84302
Kaggle	0.74961	0.84081	0.72009	0.83023
Profanity	0.68617	**0.85194**	0.66230	0.80581
Avg Ensemble	**0.81219**	0.84795	0.79086	**0.86047**

Table 5.3: Results of individual models for Dev Set, reported macro-F1, precision, recall and accuracy for each model. We show the best model in bold

5.2 Dev Error Analysis

Table 5.3 details the classification results of each model independently. We report the validation scores (OLID Test) and the test scores (SOLID Test). In addition to macro-F1, we also also included macro precision (P), recall (R) and accuracy (acc.). Figure 5.1 shows the confusion matrices for individual models in the Dev set. Here the positive class represents that the text is offensive and the negative class represents non-offensive.

It's interesting to see that OLID and SOLID models have the best macro-F1 scores out of the 6 individual models. OLID model has the best macro-recall, followed by models created by SOLID subsets as well as SOLID full set. This means those models are better at minimising type II errors, i.e. having offensive text mis-classified as being non-offensive. This is evident when looking at the confusion matrices, which show OLID has proportionally smaller number of False Negatives (54) to True Positives (186) than Profanity Check, where it has 159 False Negatives and 81 True Positives.

Conversely, model trained with Profanity Check have better precision thus better at reducing thus type I error (False Positives Rate). For instance, The Profanity Check model has 8 False Positives and 81 True Positives while OLID has 81 False Positives and 186 True Positives.

Contrary to our initial hypothesis, the models trained from under-sampling the uncertain partition of SOLID did not perform as well as the model built using the whole set in terms of macro-F1 scores. We posit that further tuning of the optimal threshold μ and σ for under-sampling, may influence the results. Possibly apply a more aggressive under-sampling could increase precision scores. Due to time constraints we leave further exploration as future work.

As each model has different strengths, we decided to experiment with various ensemble techniques to create a final model. Instead of combining all the data into one single dataset for training, we wanted to have more flexibility in tuning the impact of using each dataset, so decided to create separate models using each dataset so the final model can benefit from having larger datasets without smaller datasets being undermined.

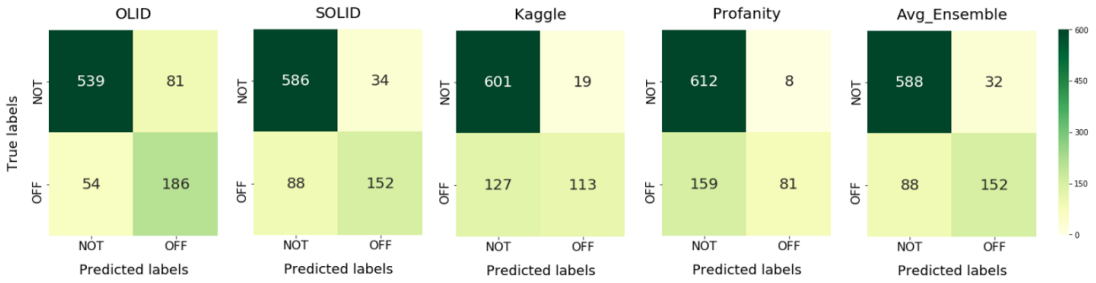

Figure 5.1: Confusion Matrices on the Dev set

5.3 Ensemble Methods

We used the output of the following models as input features to the ensemble model. The models from SOLID subsets are not included as they did not out-perform SOLID full set.

- OLID
- SOLID
- Kaggle
- Profanity

Here we experiment with different ensemble techniques, and train the ensemble model using the OLID Test dataset, with 5 fold cross validation, and 10% of dataset reserved for comparison between models.

Ensemble Method	Parameter Search	Parameter Chosen
average ensemble	N/A	equal weights of 0.25
weighted ensemble	check weights between 0 and 1 in incremental steps of 0.1	w(olid) = 0.5, w(solid) = 0.3, w(kaggle) = 0.5, w(prof) = 0.3
gradient boosting	n_estimators: [5, 10, 30, 50]	n_estimators = 10
ada boost	n_estimators: [5, 10, 30, 50]	n_estimators = 10
svm - rbf kernel	C : [0.1, 1, 10, 100] gamma: [0.01, 0.1, 1, 10]	C = 100 gamma = 0.1
svm - linear kernel	C : [0.1, 1, 10, 100]	C = 0.1
logistic regression	C : [0.1, 1, 10, 100]	C = 0.1

Table 5.4: Parameters of Ensembles, where *"n_estimators"* represents the maximum number of estimators at which boosting is terminated, *"C"* indicates the inverse of regularization strength and *"gamma"* represents the inverse of the radius of influence of samples selected by the model as support vectors

5.3.1 Average Ensemble

This is the simplest method where each individual model contribute equally to the final prediction. As there are 4 individual models, each model has a weight of 0.25. A threshold of 0.5 is then applied to the weighted sum to determine between "NON" and "OFF" for the final prediction.

5.3.2 Weighted Ensemble

Here we search for the weights of each individual model from between 0 and 1, in increments of 0.1. Again, a threshold of 0.5 is then applied to the weighted sum to determine between "NON" and "OFF" for the final prediction.

5.3.3 AdaBoost Ensemble

In AdaBoost[17] algorithm uses a series of decision trees. The first tree is trained with all observations having equal weight. After evaluating the first tree, observations in error are assigned higher weights than the correct observations when training the second tree. This process repeats for a specified number of trees as subsequent trees help to classify observations that were mis-classified by previous trees. The model is the weighted sum of predictions of all the trees, with more weight placed on stronger learners.

With individual model outputs as input features, we trained Scikit-learn's AdaBoostClassifier using default settings, with 5-fold k-fold validation combined with grid-search of hyper-parameter - *n_estimators* to find the optimal number of estimators to use for the ensemble.

5.3.4 Gradient Boosting Ensemble

A Gradient Boosting Ensemble[35] uses a differentiable loss function, a series of weak learners (decision trees) and an additive component. The decision trees used are regression trees, outputting real values. Trees are added to the model to reduce the calculated loss using gradient descent. The new tree´s output is added to the output of the previous trees, and the process repeats until a specified number of trees are reached or when the loss falls below a certain threshold.

With individual model outputs as input features, we trained Scikit-learn's GradientBoostingClassifier using default settings, with 5-fold k-fold validation combined with grid-search of hyper-parameter - *n_estimators* to find the optimal number of estimators to use for the ensemble.

5.3.5 SVM (Linear Kernel) Ensemble

We used Support Vector Machine[51] with Linear Kernel to combine individual model outputs and generate a prediction. Support Vector Machines tries to find a hyper-plane to separate data points representing different classes, with a largest possible margin, i.e. distance to its nearest data point on each side, while lowering the rate of mis-classification. The C parameter controls the amount of regularization, where a large C will cause the optimizer to choose a smaller-margin hyper-plane, while a small C value would result in a larger margin even if the hyper-plane mis-classifies some points. For our experiment, we used a grid search to find the optimal hyperparameter for C. (Table 5.4)

5.3.6 SVM (RBF Kernel) Ensemble

We also used Support Vector Machine with Radial Kernel which utilises a Radial Basis Function (RBF) to create a bell-shaped decision boundary, so that it can work with datasets that are not linearly separable. In addition to the hyperparameter C, gamma (*gamma*) is also used for regularization. A smaller *gamma* results in a smoother decision boundary while a larger gamma results in a more irregular decision boundary. Table 5.4 shows the values of C and *gamma* we used in the grid search when combining results of individual models.

5.3.7 Logistic Regression Ensemble

Lastly we used a logistic regression model to combine the results of individual models. Here the hyperparameter C is the inverse regularization strength such that higher values of C represent less regularization. We used grid search over the values of [0.1, 1, 10, 100] for C to find the best hyperparameter for the ensemble.

5.4 Ensembles Results On Dev Set

Models	macro-F1	macro-P	macro-R	acc.
Logistic Regression	0.80784	0.84698	0.78542	0.85814
AdaBoost	0.80848	0.85073	0.78495	0.85930
SVM linear	0.80967	0.84332	0.78925	0.85814
Avg	0.81219	0.84795	0.79086	0.86047
Gradient Boost	0.81414	**0.85400**	0.79120	**0.86279**
SVM RBF	0.81474	0.85269	0.79247	**0.86279**
Weighted	**0.81528**	0.80540	**0.82890**	0.84535

Table 5.5: Results of ensembles, reported macro F1, precision (P), recall (R) and accuracy (acc.) for Dev set. Bold values show the best performing models

We report the model ensemble results on the dev set (OLID Test) in Table 5.5. Weighted Ensemble achieved the highest macro-F1 score, followed by SVM RBF model, Gradient Boost Model and Average Ensemble.

While searching for the best weights using Grid Search, we found over 1000 combinations that resulted in the same top score. This suggests that the amount of training data is insufficient and we may be over-fitting to the dataset. This result also suggest potential issues with more complex ensemble techniques.

5.4.1 Final Model

Model	p value
Weighted	< 0.01
Logistic Regression	> 0.01
SVM RBF	> 0.01
SVM Linear	> 0.01
AdaBoost	> 0.01
Gradient Boost	> 0.01

Table 5.6: Results of Wilcoxon Rank-Sum Test - comparing prediction distribution of different ensembles against predictions by the average weighted ensemble

We applied the Wilcox-on Rank-Sum Test to compare the prediction distributions of each ensemble against the predictions by the average ensemble for the dev set (Table 5.6). Only the weighted ensemble produced a result that is statistically different ($p<0.01$). Ultimately we decided to submit the Avg ensemble since it is a simple model that is less prone to over-fitting and improved upon models trained with individual datasets.

5.5 Test Results

The official Test dataset, SOLID Test, contains 3,887 tweets. Of which, 1,080 tweets (28%) are offensive. (Table 5.7). Like the training and dev datasets, the test dataset is also unbalanced.

Class	SOLID Test
NON	2,807 (72%)
OFF	1,080 (28%)
TOTAL	3,887 (100%)

Table 5.7: Target Distribution of Test Dataset.

Table 5.8 and Table 5.9 show results of our individual models and ensemble models on the test dataset. We have submitted the Average Ensemble as the final model, and it is fortunate that the model in fact out-performed all other ensemble models as well as all the individual models. The confusion matrices on the test set (Figure 5.2) highlights that the Avg Ensemble was a good all-rounder compared to the individual models, with the second lowest False Negatives rate and moderate False Positive rates. Table 5.8 shows that the Avg Ensemble has the best macro-Recall and macro-F1 in comparison to individual models, and Table 5.9 shows the Avg Ensemble has the highest score across all matrices for ensemble models.

Our team, M20170548, achieved a macro-F1 score of 0.91344 in the OffensEval shared task for English sub-task A, and ranked 11th out of 81 teams. The full teams results are shown in Table 5.10. We note that the scores are very close, where the top 20 teams are within 0.01 range, suggesting a very tight competition.

Model	macro-F1	macro-P	macro-R	acc.
OLID	0.90428	0.88543	0.93798	0.91742
SOLID	**0.91136**	0.89223	**0.94510**	0.92359
SOLID A	0.90822	0.88877	0.94439	0.92050
SOLID B	0.90814	0.88877	0.94382	0.92050
Kaggle	0.90998	**0.90436**	0.91619	**0.92668**
Profanity	0.85402	0.89636	0.82852	0.89220
Avg Ensemble	**0.91344**	0.89464	**0.94539**	0.92560

Table 5.8: Results of individual models for Test Set, reported macro-F1, precision, recall and accuracy for each model. We show the best model in bold

Models	macro-F1	macro-P	macro-R	acc.
Logistic Regression	0.91212	0.89358	0.94325	0.92462
AdaBoost	0.91304	0.89442	0.94436	0.92539
SVM linear	0.91136	0.89223	0.94510	0.92359
Avg	**0.91344**	**0.89464**	**0.94539**	**0.92565**
Gradient Boost	0.91244	0.89385	0.94371	0.92488
SVM RBF	0.91076	0.89208	0.94265	0.92333
Weighted	0.90521	0.88597	0.94090	0.91700

Table 5.9: Results of ensemble models for Test Set, reported macro-F1, precision, recall and accuracy for each model. We show the best model in bold

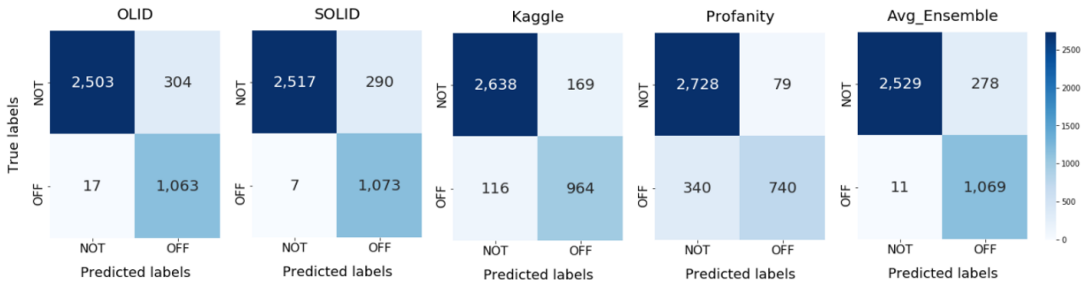

Figure 5.2: Confusion Matrices on the Test set

	Team	Score	#	Team	Score	#	Team	Score
1	UHH-LT	0.9204	29	UTFPR	0.9094	57	OffensSzeged	0.9032
2	Galileo	0.9198	30	IU-UM@LING	0.9094	58	aprosio	0.9032
3	Rouges	0.9187	31	talhaanwar	0.9093	59	RGCL	0.9006
4	GUIR	0.9166	32	SSN NLP	0.9092	60	byteam	0.8994
5	KS@LTH	0.9162	33	Hitachi	0.9091	61	jmperez	0.899
6	kungfupanda	0.9151	34	kathrync	0.9091	62	PUM	0.8973
7	TysonYU	0.9146	35	XD	0.909	63	shardul007	0.8927
8	AlexU-BackTranslation-TL	0.9139	36	UoB	0.909	64	I2C	0.8919
9	SpurthiAH	0.9136	37	PAI-NLP	0.9089	65	sonal.kumari	0.89
10	amsqr	0.9135	38	PingANPAI	0.9089	66	IJS	0.8887
11	***m20170548***	***0.9134***	39	VerifiedXiaoPAI	0.9089	67	IR3218	0.8843
12	Coffee Latte	0.9132	40	nlpUP	0.9089	68	TeamKGP	0.8822
13	wac81	0.9129	41	NLP Passau	0.9088	69	UNT Linguistics	0.882
14	hwijeen	0.9129	42	TheNorth	0.9087	70	janecek1	0.8744
15	UJNLP	0.9128	43	problemConquero	0.9085	71	Team Oulu	0.8655
16	ARA	0.9119	44	Lee	0.9084	72	TECHSSN	0.8655
17	Ferryman	0.9115	45	Wu427	0.9081	73	KDELAB	0.8653
18	ALT	0.9114	46	ITNLP	0.9081	74	HateLab	0.8617
19	SINAI	0.9105	47	Better Place	0.9077	75	IASBS	0.8577
20	MindCoders	0.9105	48	IIITG-ADBU	0.9075	76	IUST	0.8288
21	IRLab DAIICT	0.9104	49	‘doxaAI	0.9075	77	Duluth	0.7714
22	erfan	0.9103	50	NTU Place	0.9067	78	RTNLU	0.7665
23	Light	0.9103	51	FERMI	0.9065	79	KarthikaS	0.6351
24	KAFK	0.9099	52	mdherath	0.9063	80	Bodensee	0.4954
25	PALI	0.9098	53	INGEOTEC	0.9061	81	Majority Baseline	0.4193
26	PRHLT-UPV	0.9097	54	PGSG	0.906	82	IRlab@IITV	0.0728
27	YNU oxz	0.9097	55	SRIB2020	0.9048			
28	IITP-AINLPML	0.9094	56	tcaselli	0.9036			

Table 5.10: Results for OffensEval 2020 English Subtask A. Teams are ranked in decreasing order of macro-averaged F - From [61]

Chapter 6

Categorization of Offense Types (Level B)

6.1 Training Process

We created a Training and a Dev set by splitting the OLID training set using the proportion shown in Table 6.1.

Class	Training Set	Dev Set	Combined
TIN	3487 (88%)	389 (88%)	3,876 (88%)
UNT	473 (12%)	51 (12%)	524 (12%)
TOTAL	3960 (100%)	440 (100%)	4400 (100%)

Table 6.1: Target Distribution of Training and Dev Set

Using a pre-trained "BERT-base-uncased" BERTForSequenceClassification model from PyTorch-Transformer library, we compared the results by adjusting the following hyperparameters:

- number of training epochs
- learning rate for AdamW optimizer
- number of warm up steps for the Warmup Linear steps
- train batch size

6.2 Dev Results

The model achieved the results shown in Table 6.2 on the Dev set when trained over 4 epochs. We logged the loss and accuracy and macro-F1 for each epoch, and noticed that although the loss was the lowest in epoch 4, epoch 3 achieved a higher macro-F1.

macro-F1	macro-P	macro-R	Accuracy
0.60163	0.66823	0.58132	0.87727

Table 6.2: Results on the Dev set

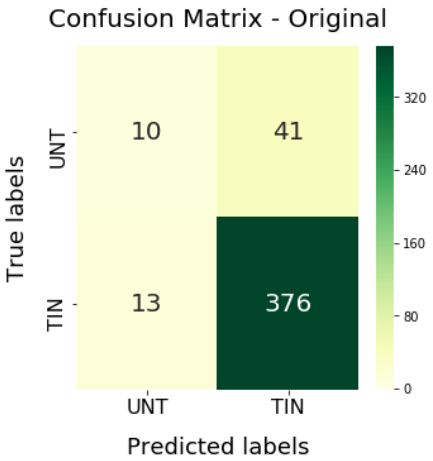

Figure 6.1: Confusion Matrix on the Dev set

6.3 Error Analysis

The confusion matrix in Figure 6.1 shows that out of 51 total Untargeted offences (UNT), 41 of them (80%) have been mis-classified as Targeted (TIN). This is perhaps not unexpected, given that the data is highly unbalanced and the classifier is prone to predict more majority class (TIN).

To mitigate this, We experimented with changing the classification threshold from a default at 0.50 to 0.95 in increments of 0.05. Table 6.3 outline the results at each step. We noticed that macro-Precision and Accuracy was the highest at a threshold of 0.50 and generally decreases as the threshold is increased. Conversely macro-Recall increases as the threshold is increased.

Threshold	macro-F1	macro-P	macro-R	Accuracy
0.50	0.60164	**0.66823**	0.58133	**0.87727**
0.55	0.59917	0.65905	0.58004	0.87500
0.60	0.60861	0.66322	0.58856	0.87500
0.65	0.60861	0.66323	0.58856	0.87500
0.70	0.60861	0.66323	0.58856	0.87500
0.75	0.60611	0.65528	0.58728	0.87273
0.80	0.59888	0.63455	0.58342	0.86591
0.85	0.60517	0.63391	0.59065	0.86364
0.90	**0.61632**	0.63335	0.60512	0.85909
0.95	0.60733	0.60205	**0.61397**	0.82955

Table 6.3: Dev Results for models with different classification thresholds, reported macro F1, precision (P), recall (R) and accuracy (acc.). Bold values show the best performing models

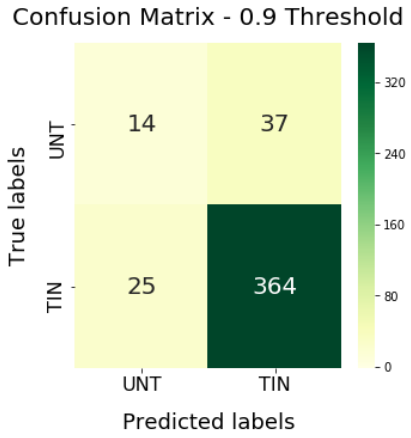

Figure 6.2: Confusion Matrix on Dev Set with modified Threshold

Intuitively this makes sense, as the threshold gets higher, there will be less instances of minority class (UNT) being mis-classified as majority class (TIN), thus increasing the Recall for the minority class (UNT). This comes at the expense of Recall for the majority class, as model will mis-classify some majority class instances (TIN) as being of minority class (UNT). As macro-Recall is the unweighted mean of Recall from both classes and relative improvement of Recall in minority class is greater than the relative degradation in the majority class, macro-Recall increases as the threshold is moved higher. There could be a point where this is no longer the case but in our scenario this observation holds true.

Let´s consider the majority class (TIN) as being the Positive class and the minority class (UNT) as the Negative class.

As the classification threshold increases and more instances are being pushed to the minority class, it reduces the macro-Precision scores. This happens as the number of False Negatives increases faster than the number of True Negatives, since the Positive class (TIN) is also the majority and more of the previously predicted Positive instances will became Negative as the threshold changes. Precision measures the ratio of correctly classified instance of the class to the total number predicted for that class and as a result, Precision for the minority class decreases as the threshold changes.

Finally, Table 6.3 shows that the classification threshold of 0.90 produced the highest macro-F1 score, although the threshold of 0.50 had a higher accuracy and macro-Precision. The confusion matrix on Figure 6.2 represents the result where the threshold was changed to 0.9. Compared to Figure 6.1, the number of False Positives has been reduced from 41 to 37, but the number of False Negatives has increased from 13 to 25. The macro-F1 score is increased as the amount of improvement of the minority class is greater than the degradation of the majority class.

Table 7.4 shows all the tweets with changed predictions when the threshold was changed from 0.5 to 0.9. There were 25 in total, with predictions changed from Targeted to Untargeted. According to the original label only 7 of them were correct in

this change. However, examining the content suggests majority of them are in fact untargeted, which makes us question the accuracy of the original annotation.

actual label	original prediction	adjusted prediction	tweet
0	1	0	Good- that soundtrack you DMed me with was fire as fuck and I still listen to it often Bad - My tracks are better (joking I'm shit)
0	1	0	Don't ever stop lil ma *COUGH* "eddandflow" *COUGH* oh shit is that my Snapchat how'd that slip out?! ..add me
0	1	0	Fuck yeah!!
0	1	0	It's going to suck with any rating. Venom without the Spiderman origin isn't really Venom. This is just a Sony cash grab.
0	1	0	Gun control now! Enough is enough!
0	1	0	Josh I am not fucking a loli robot
0	1	0	House of flowers!! Shits crazy
1	1	0	REST IN PUSSY
1	1	0	WHAT THE FUCK YOURE SO GORGEOUS IM ACTUALLY SOBBING
1	1	0	A Long Ass Blog That Might Not Make Sense
1	1	0	and in the are going to get a little bit riled up me thinks?!
1	1	0	Action talks and BULLSHIT walks
1	1	0	yea shit sucks thats why i stopped watching anime mostly
1	1	0	(She is clearly agitated in a defensive stance) Shit- what is she doing here
1	1	0	Fuuckk youu
1	1	0	I would name my dogs breed but I have no fucking clue what he is
1	1	0	Boycotting NFL for the rest of my life.
1	1	0	Clown
1	1	0	achichincle lamebotas!
1	1	0	Yep. Sure do. And he is been proving it on a daily basis.
1	1	0	real shit
1	1	0	Evolution. An evolution we helped to enact. is economic and social suicide.
1	1	0	I think it is more of a choice for ppl to stay hooked on shit

Table 6.4: Examples of prediction changes after threshold adjustment, where 1 represents Targeted (TIN) and 0 represents Untargeted (UNT)

6.4 Final Model

We decided to use a classification threshold of 0.9 instead of 0.5 for the final model in order to improve model performance on the minor class. The final model was trained by combining the training and dev set and using the following hyperparameters:

- Number of Training Epochs = 3
- Learning Rate = 5e-5
- Train Batch Size = 16
- Optimizer Warm up steps = 10% of total steps

6.5 Test Results

The target distribution of the test dataset (Table 6.5) is similar to the training dataset (Table 6.1) and is also highly unbalanced.

Table 6.6 shows results of the Test set, where the model with adjusted threshold of 0.9 improved upon the original model with threshold of 0.5 in terms of macro-F1 score. This was achieved by an increase in Recall of the minor class (UNT) at the expense of a slight degradation of Recall in the major class (TIN) and also the Precision and Accuracy scores. This is illustrated by the confusion matrices in Figure 6.3, where the original model only identified 9 (33%) out of 27 Untargeted offenses while the adjusted model correctly identified 13 (48%).

Interestingly a threshold of 0.9 resulted in 26 instances being classified as Untargeted, which is quite close to the actual number of 27 Untargeted offenses.

Although we did not officially participate OffensEval 2019, the adjusted model achieved a good result. Table 6.7 shows the official results of the task. A macro-F1 score of 0.71367, which would place it 5th out of 76 participating teams in the task.

Class	OLID Test
TIN	213 (89%)
UNT	27 (11%)
TOTAL	240 (100%)

Table 6.5: Target Distribution of Test Dataset

model	macro-F1	macro-P	macro-R	Accuracy
original	0.69332	**0.78161**	0.65493	**0.90417**
adjusted	**0.71367**	0.71729	**0.71022**	0.88750

Table 6.6: Results on the Test set, reported macro F1, precision (P), recall (R) and accuracy (acc.). Bold values show the best performing models

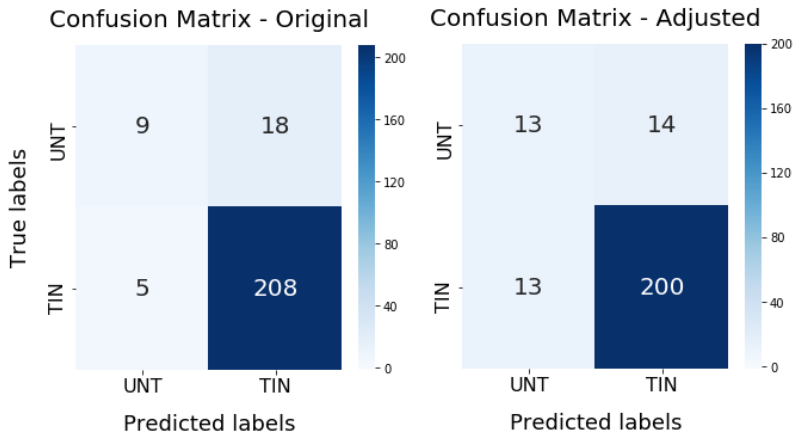

Figure 6.3: Confusion Matrices on the Test set

Sub-task B	
Team Ranks	**F1 Range**
1	0.755
2	0.739
3	0.719
4	0.716
5	0.708
6	0.706
7	0.700
8	0.695
9	0.692
10	0.687
11-14	.680-.682
15-24	.660-.671
25-29	.640-.655
30-38	.600-.638
39-49	.553-.595
50-62	.500-.546
63-74	.418-.486
75	0.270
76	0.121

Table 6.7: OffensEval 2019 Sub-task B results: F1-Macro for top-10 teams followed by the rest of the teams grouped in ranges. From [60]

Chapter 7

Offense Target Identification (Level C)

7.1 Training Process

We again created a Training and a Dev set by splitting the OLID training set using the proportion shown in Table 7.1.

Class	Training Set	Dev Set	Combined
IND	2,163 (62%)	244 (63%)	2,407 (62%)
GRP	970 (28%)	104 (27%)	1,074 (28%)
OTH	355 (10%)	40 (10%)	395 (10%)
TOTAL	3,488 (100%)	388 (100%)	3,876 (100%)

Table 7.1: Target Distribution of Training and Dev Set

Using a pre-trained "BERT-base-uncased" BERTForSequenceClassification model from PyTorch-Transformer library, we compared the results by adjusting the following hyperparameters:

- number of training epochs
- learning rate for AdamW optimizer
- number of warm up steps for the Warmup Linear steps
- train batch size

The target distributions are unbalanced, and we adjusted the classification threshold to improve the macro-F1 score.

This time we applied the thresholds used by the task´s top performing team, Nikolov-Radivchev[39], giving preferences to the minority classes OTH and GRP by lowering their classification thresholds and evaluating in the following order:

- if probability for OTH exceeds 0.2, assign as OTH
- else if probability for GRP exceeds 0.3, assign as GRP
- otherwise assign as IND

7.2 Dev Results

model	macro-F1	macro-P	macro-R	Accuracy
original	0.551911	**0.659015**	0.573676	**0.76804**
adjusted	**0.579067**	0.573554	**0.590111**	0.72165

Table 7.2: Results of original model and the model with adjusted thresholds, showing macro F1, precision, recall and accuracy for the Dev sets. Bold values show the best performing model

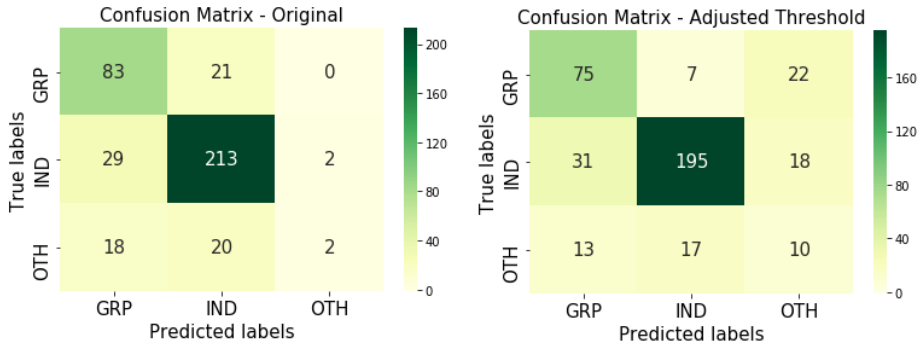

Figure 7.1: Confusion Matrices on the Dev set for the original model and the model with adjusted threshold

actual label	original prediction	adjusted prediction	count
GRP	GRP	OTH	15
	IND	GRP	7
		OTH	7
IND	GRP	OTH	9
	IND	GRP	11
		OTH	7
OTH	GRP	OTH	7
	IND	GRP	2
		OTH	1
TOTAL			**66**

Table 7.3: Count of prediction changes as a result of threshold adjustment

7.3 Error Analysis

actual label	original prediction	adjusted prediction	tweet
GRP	GRP	OTH	@USER @USER Gun control isn't about public safety...it's about hating 'rednecks' and social engineering.
GRP	GRP	OTH	@USER @USER @USER URL The left party relativizing and playing down communist crimes
GRP	GRP	OTH	@USER Antifa is Israel's thug squad in America
GRP	GRP	OTH	@USER I'm just here to see which liberals decide to politicize this tragedy (despite Cincy having tight gun control)...
GRP	IND	GRP	@USER I am definitely for antifa members getting themselves killed.
GRP	IND	OTH	@USER @USER The more Antifa BS average Americans see in the news the more repulsed they are by their antics.
IND	GRP	OTH	Agreed great idea! Wish we could also remove her from existence! #MAGA #PatriotsUnited #ProLife #ProIsrael #WeAreQ. URL
IND	GRP	OTH	@USER It's Good for our Gun Business. More Guns sell when these people bring up Gun Control. Keep up the Great Work. Guns for All.
IND	GRP	OTH	Democrat can't even compete w @USER #MAGA URL
IND	IND	GRP	@USER @USER @USER Yes he is and the democrats hate it
OTH	GRP	OTH	@USER @USER I think the pope and some others should be prosecuted for covering up and protecting child rapists!
OTH	GRP	OTH	@USER Religious persecution is bad... ...when it's your religion being persecuted. URL
OTH	GRP	OTH	@USER @USER @USER @USER Name the social injustices caused by the GOP and the death of anyone caused by members of it I dare you. How many people have lost their lives to the idiocy of gun control?
OTH	GRP	OTH	#PeoplesVote #StopBrexit @USER #remain I don't want to go.. either, You know what, After we leave, My slogans going to be It used to be better than this"should have stayed in. Useless goverment. URL
OTH	IND	GRP	@USER @USER @USER Haha you liberals are so desperate it's hilarious. Trying at all ends to set him up. This BS last minute stalling tactic about the alleged HS incident and the bait of emails that were sent to him. And people wonder how Trump won? Because they're tired of Democrats and their c

Table 7.4: Examples of prediction changes after threshold adjustment

Looking at Table 7.2 we can see that the threshold adjustments improved macro-F1 and macro-Recall, just as they did in previously in Chapter 6. The confusion matrices on Figure 7.1 shows a big change in the number of predictions for OTH between the original model and one with the adjusted threshold, increasing from 4 to 50. The actual number of OTH in the Dev set is 40, so the adjusted model has a closer count, even though only 10 out of those 50 are correct. However by making the model more sensitive to the minority classes, it increases the Recall for those classes and ultimately improves the macro-F1 score as well.

Table 7.3 shows number of prediction changes as a result of the threshold adjustments. There are a total of 66 predication changes, with 31 (15 + 9 + 7) predictions changing from GRP to OTH and 20 (7 + 11 + 2) predictions changed from IND to GRP. Of these, only 7 out of 31 (23%) for the first group, and 7 out of 20 (35%) for the second group were correct.

Table 7.4 shows samples of the prediction changes in each category. Firstly we can see a high proportion of tweets relating to politics, spanning issues such as gun control, antifa, MAGA (Make America Great Again) and brexit. Generally these should be classified as GRP according to the annotation guideline, but in fact it is not always obvious or consistently labelled. For instance, "*@USER It's Good for our Gun Business.*

More Guns sell when these people bring up Gun Control. Keep up the Great Work. Guns for All." has a label of IND but perhaps should be labelled as GRP, and similarly for "*@USER @USER @USER Yes he is and the democrats hate it*" as offense is targeted towards "democrats" rather than the individual.

Secondly it appears that the label OTH can be ambiguous, sometimes used instead of GRP - "*@USER @USER @USER @USER Name the social injustices caused by the GOP and the death of anyone caused by members of it I dare you. How many people have lost their lives to the idiocy of gun control?*" and sometimes when the target is unclear - "*@USER Religious persecution is bad... ...when it's your religion being persecuted. URL*". It can be argued that the sentence is not targeted or offensive in the first place.

The samples highlights the difficulty in this task, and in offensive language classification in general, where the interpretation of offensiveness can be highly subjective.

7.4 Final Model

The final model was trained by combining the training and dev set and using the following hyperparameters:

- Number of Training Epochs = 3
- Learning Rate = 2e-5
- Train Batch Size = 16
- Optimizer Warm up steps = 5% of total steps

We also adjusted the model´s classification thresholds as described earlier.

7.5 Test Results

Table 7.5 shows the target distribution of the Test dataset. Compared to the Training and Dev datasets (Table 7.1), the Test dataset is slightly less unbalanced.

Class	OLID Test
IND	100 (47%)
GRP	78 (37%)
OTH	35 (16%)
TOTAL	213 (100%)

Table 7.5: Target Distribution of Test Dataset

Our model with the adjusted thresholds has significantly improved its performance in all aspects except for Accuracy (Table 7.1). This is likely due to the increase in the number of OTH being classified. The original model only has 1 instance classified as OTH while the adjusted model has 24 instances of OTH. Comparing this with an actual OTH count of 35, the adjusted model has a closer total of classifying OTH, although

only 11 out of the 24 are correct. This also suggests the difficulty in identifying a minority class in an imbalanced dataset.

We did not officially participate in OffensEval 2019, but by adapting the threshold used by the top team, our model achieved a macro-F1 score of 0.643352, ranking it number 2. Table 7.7 shows the official results of the task. We note that the general scores are not very high, perhaps due to the imbalance of data, and some inconsistency in the annotation.

model	macro-F1	Precision	Recall	Accuracy
original	0.530278	0.487709	0.582051	**0.732394**
adjusted	**0.643352**	**0.653735**	**0.641172**	0.727700

Table 7.6: Results of original model and the model with adjusted thresholds. Bold values show the best performing model

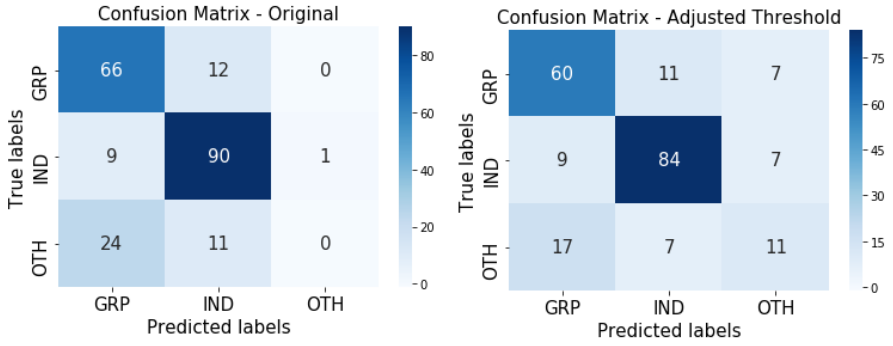

Figure 7.2: Confusion Matrices on the Test set for the original model and the model with adjusted threshold

Sub-task C	
Team Ranks	**F1 Ranage**
1	0.66
2	0.628
3	0.626
4	0.621
5	0.613
6	0.613
7	0.591
8	0.588
9	0.587
10	0.586
11-14	.571-.580
15-18	.560-.569
19-23	.547-.557
24-29	.523-.535
30-33	.511-.515
34-40	.500-.509
41-47	.480-.490
46-60	.401-.476
61-65	.249-.340

Table 7.7: OffensEval 2019 Sub-task C results: F1-Macro for top-10 teams followed by the rest of the teams grouped in ranges. From [60]

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis explores the topic of Offensive Language Classification in Social Media in the context of research tasks in OffensEval 2020 and OffensEval 2019. We addressed the topic in three levels:

Level A - Offensive Language Detection - ***is the tweet offensive?***

The first question corresponds to subtask A of OffensEval 2020 and is satisfactorily answered in this context as our model achieved a reasonable performance with a macro-F1 score of 0.9134, placing it 11th out of 82 submissions.

The main challenge was due to semi-supervised nature of the official dataset, SOLID, where the annotations were less reliable than a manually annotated one and as a result required further interpretation. This limitation was overcame by including additional offensive language classification datasets with similar annotation guideline that were manually labelled, as well as using tools that cover the shortcomings of the original dataset such as the profanity check library. Rather than creating one single combined dataset, we fine-tuned separate BERT models for each different dataset and experimented with various ensemble techniques to combine their prediction labels. This allowed for more flexibility in tuning the impact of each dataset and allowed us to get the benefit of larger datasets without undermining smaller ones. Combining the output labels of the models instead of their probability scores allows us to deal with the same classification space and bypass potential issue in the variance of their estimation margins.

BERT architecture utilises contextualized embedding and multi-headed attention to acquire deep knowledge of the language it models. By fine-tuning pre-trained BERT models, Our final model performed better in Test compared to Dev, proving it did not

over-fit to the dataset and that BERT is a suitable architecture for this task.

Level B - Categorisation of Offense Type - ***is the offense is targeted?***

The second question corresponds to subtask B of OffensEval 2019 and warrants further investigation. The existing top five submissions for subtask B achieved micro-F1 scores in the range of 0.708 to 0.755. our model performed well in context of the task, scoring 0.714, equivalent to the 5th place, but there is room for improvement.

Similar to subtask A, we fine-tuned a pre-trained BERT model, BERT-base-uncased. The training dataset contained roughly roughly 7 times more targeted tweets compared to untargeted tweets, causing the model to have a low recall on the minority class. By experimenting with the classification threshold to optimize the macro-F1 score, we finalised on a model with a classification threshold of 0.90 for the majority class.

The challenge in this task was multi-fold - the dataset is relatively small and unbalanced, and in addition appear to have inconsistent annotation. The syntactic structure of the sentence may be important in the task of distinguishing between targeted offense and untargeted offense. We note the top teams in this task utilized a rule-based, dictionary approach to calculate the offensive score of a sentence as a linear combination of individual words offensiveness and an "intensifier" factor of words syntactically related to the word. The idea is interesting and worth exploring.

Level C - Offensive Target Identification - ***if the offense is targeted identify the target as one of the following - individual, group or others***

The third question corresponds to subtask C of OffensEval 2019. The top five submissions for subtask C achieved micro-F1 scores in the range between 0.613 and 0.660. Our model scored 0.643, equivalent to the 2nd place, but we believe more research is required in this area.

The dataset for this task was highly unbalanced, with 62% labelled as "IND", 28% as "GRP" and 10% as "OTH". We adjusted the classification thresholds to increase the recall of GRP and OTH predictions, which significantly improve the macro-F1 score in both Dev and Test.

This appears to be a difficult task as the macro-F1 scores achieved by teams in the top 50% percentile ranged between 0.511 and 0.660. Like subtask B, this task also suffers from having a small and unbalanced dataset. The low scores could also be attributed to the ambiguity between the labels GRP and OTH. By task definition, the GRP label includes posts targeting groups of people with similar backgrounds or political and religious beliefs while the OTH label applies for posts not belonging to either IND or GRP, such as organisation, situation, event or issue. However a political issue, such as gun control or Brexit, are sometimes inconsistently annotated. Likewise when it comes to annotating offensive post related to politicians, the examples in the training dataset are inconsistent.

8.2 Future Work

As future work, we propose that Offensive Language Classification in Social Media can be improved by focusing on these areas:

Data Quality

Our error analysis identified scenarios with inconsistent annotation, suggesting a need for more detailed guidelines focusing on nuanced cases, provision of explicit examples on how to annotate and increase the criteria for inter-annotator agreement.

For subtask A, our experiment in under-sampling uncertain regions of SOLID for training did not yield a superior result. We leave as future work more aggressive under-sampling schemes to access the consequences of only training on highly confident predictions, and differ additional semi-supervised strategies to improve results using SOLID dataset.

Social media platforms differ in their offensive language classification guidelines, and one could divide the training data into different regions, with a standardised set featuring regions with high agreement, and a separate "tricky" set that is annotated per individual platform's requirement. The model can then be fine-tuned by tailoring the content and increasing the size of the tricky set.

Data Quantity

We have shown that model performance can be improved by combining models trained on different datasets with similar annotation guidelines in subtask A. This approach can be extended to include additional publicly available datasets in the future. Data augmentation techniques such as back translation, or translating datasets from a different language may also help to supplement cases where data quantity is insufficient.

For subtask B and C, potential future work is to combine SOLID from OffensEval 2020 and OLID from OffensEval 2019, and explore with ensemble and sub-sampling techniques as we did for subtask A.

If industry and the research community can agree on a common schema, such as OLID, then future research can leverage off increased data quantity as a result of data sharing. As a starting point we propose to extend the OLID schema to differentiate between implicit and explicit abuse in order to incorporate research in the area of micro-aggression.

Model Architectures

As NLP continue to accelerate and evolve, further gains can be made by applying the new architectures in this domain. For instance, one could explore and compare performances of various transformer models like XLNet [58] and BERT variants such RoBERTa [32], DistilBERT [47]. Prior work has shown benefits in creating ensembles from transformer based architectures as well as in combination with other deep learning models, such as CNN [30] and LSTM [25].

Techniques such as multi-task [11] learning and label transfer network [1] may be useful when task specific data is limited. Furthermore, as requirement for offensive language classification vary, few-shot learning may provide a useful alternative for tailoring the system to detect specific scenarios when the amount of data is limited [49].

Finally, for subtask B and C, the inclusion of entity embeddings (NER) [34] as an additional feature into the final classification layer may assist the classifier in differentiating between targeted and untargeted posts, as well as identifying the type of target if one exists.

Bibliography

[1] I. Augenstein, S. Ruder, and A. Søgaard. "Multi-task Learning of Pairwise Sequence Classification Tasks Over Disparate Label Spaces." In: *NAACL-HLT*. 2018.

[2] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." In: *CoRR* abs/1409.0473 (2015).

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. "Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling." In: *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 1992, pp. 161–167.

[4] BBC. "Christchurch shootings: Social media races to stop attack footage." In: (2019). URL: https://www.bbc.com/news/technology-47583393.

[5] BBC. "Hana Kimura: Netflix star and Japanese wrestler dies at 22." In: (2020). URL: https://www.bbc.com/news/world-asia-52782235.

[6] BBC. "Regulator Ofcom to have more powers over UK social media." In: (2020). URL: https://www.bbc.com/news/technology-51446665.

[7] BBC. "Twitter tags Trump tweet with fact-checking warning." In: (2020). URL: https://www.bbc.com/news/technology-52815552.

[8] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. "Neural Probabilistic Language Models." In: *Innovations in Machine Learning*. Ed. by D. E. Holmes and L. C. Jain. Vol. 194. Studies in Fuzziness and Soft Computing. Berlin/Heidelberg: Springer-Verlag, 2006, pp. 137–186. ISBN: 3-540-30609-9. DOI: 10.1007/3-540-33486-6{\textunderscore}6.

[9] D. Blei, A. Ng, M. Jordan, and J. Lafferty. "Journal of Machine Learning Research 3 (2003) 993-1022 Submitted 2/02; Published 1/03 Latent Dirichlet Allocation." In: (Feb. 2003).

[10] P. Brown and P. Della. "Class-based n-gram models of natural language." In: *Computational Linguistics* 28 (Jan. 2006), pp. 477–480.

[11] R. Caruana. "Multitask Learning." In: 1997.

[12] E. Culliford and K. Paul. "Twitter again slaps warning on Trump tweet threatening force against protesters." In: (2020). URL: https://www.reuters.com/article/us-usa-trump-twitter/twitter-again-slaps-warning-on-trump-tweet-threatening-force-against-protesters-idUSKBN23U33V.

[13] A. M. Dai and Q. V. Le. "Semi-supervised Sequence Learning." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 3079–3087. URL: http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

[14] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber. "Automated Hate Speech Detection and the Problem of Offensive Language." In: *ICWSM*. 2017.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *ArXiv* abs/1810.04805 (2019).

[16] P. Fortuna, J. Ferreira, L. Pires, G. Routar, and S. Nunes. "Merging Datasets for Aggressive Text Identification." In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 128–139. URL: https://www.aclweb.org/anthology/W18-4416.

[17] Y. Freund and R. E. Schapire. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1997.

[18] A. Geron. *Hands-On Machine Learning With Scikit-Learn And Tensorflow. Concepts, tools and techniques to build intelligent systems*. O'Reilly Media, 2017, pp. 409–412.

[19] G. Grefenstette and L. Wilber. *Search-based Applications: At the Confluence of Search and Database Technologies*. Synthesis lectures on information concepts, retrieval, and services. Morgan & Claypool Publishers, 2011. ISBN: 9781608455072. URL: https://books.google.pt/books?id=3NAt4cAwVV4C.

[20] J. Han, S. Wu, and X. Liu. "jhan014 at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 652–656. DOI: 10.18653/v1/S19-2116. URL: https://www.aclweb.org/anthology/S19-2116.

[21] D. Hand and K. Yu. "Idiot's Bayes: Not So Stupid after All?" In: *International Statistical Review* 69 (May 2007), pp. 385 –398. DOI: 10.1111/j.1751-5823.2001.tb00465.x.

[22] Z. S. Harris. "Distributional Structure." In: WORD 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520.

[23] T. Ho. "Random Decision Forest." In: *Proceedings of the 3rd International Conference on Document Analysis and Recognition*. Montréal, Canada, Aug. 1995, pp. 278–282.

[24] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9 (1997), pp. 1735–1780.

[25] S. Hochreiter and J. Schmidhuber. "Long Short-term Memory." In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[26] J. Howard and S. Ruder. "Fine-tuned Language Models for Text Classification." In: *ArXiv* abs/1801.06146 (2018).

[27] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. "Bag of Tricks for Efficient Text Classification." In: *CoRR* abs/1607.01759 (2016). arXiv: 1607.01759. URL: http://arxiv.org/abs/1607.01759.

[28] A. Karpathy. "The Unreasonable Effectiveness of Recurrent Neural Networks." In: *http://karpathy.github.io/2015/05/21/rnn-effectiveness/* (2015).

[29] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri. "Benchmarking Aggression Identification in Social Media." In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1–11. URL: https://www.aclweb.org/anthology/W18-4401.

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition." In: 1.4 (1989). ISSN: 0899-7667.

[31] P. Liu, W. Li, and L. Zou. "NULI at SemEval-2019 Task 6: Transfer Learning for Offensive Language Detection using Bidirectional Transformers." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 87–91. DOI: 10.18653/v1/S19-2011. URL: https://www.aclweb.org/anthology/S19-2011.

[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: *ArXiv* abs/1907.11692 (2019).

[33] D. Mahata, H. Zhang, K. Uppal, Y. Kumar, R. R. Shah, S. Shahid, L. Mehnaz, and S. Anand. "MIDAS at SemEval-2019 Task 6: Identifying Offensive Posts and Targeted Offense from Twitter." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 683–690. DOI: 10.18653/v1/S19-2122. URL: https://www.aclweb.org/anthology/S19-2122.

[34] E. Marsh and D. Perzanowski. "MUC-7 Evaluation of IE Technology: Overview of Results." In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. 1998. URL: https://www.aclweb.org/anthology/M98-1002.

[35] L. Mason, J. Baxter, P. L. Bartlett, and M. Frean. "Boosting Algorithms as Gradient Descent in Function Space." In: (1999). URL: http://sciencewise.info/resource/Gradient_boosting/Gradient_boosting_by_Wikipedia.

[36] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space." In: *CoRR* abs/1301.3781 (2013).

[37] S. Modha, P. Majumder, and T. Mandl. "Filtering Aggression from the Multilingual Social Media Feed." In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 199–207. URL: https://www.aclweb.org/anthology/W18-4423.

[38] A. Nikolov and V. Radivchev. "Nikolov-Radivchev at SemEval-2019 Task 6: Offensive Tweet Classification with BERT and Ensembles." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 691–695. DOI: 10.18653/v1/S19-2123. URL: https://www.aclweb.org/anthology/S19-2123.

[39] A. Nikolov and V. Radivchev. "Nikolov-Radivchev at SemEval-2019 Task 6: Offensive Tweet Classification with BERT and Ensembles." In: *SemEval@NAACL-HLT*. 2019.

[40] C. Olah. "Understanding LSTM Networks." In: *http://colah.github.io/posts/2015-08-Understanding-LSTMs/* (2015).

[41] B. A. Pearlmutter. "Learning State Space Trajectories in Recurrent Neural Networks." In: *Neural Computation* 1 (1989), pp. 263–269.

[42] J. Pennington, R. Socher, and C. Manning. "Glove: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[43] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep contextualized word representations." In: *ArXiv* abs/1802.05365 (2018).

[44] A. Radford. "Improving Language Understanding by Generative Pre-Training." In: 2018.

[45] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri, and P. Nakov. "A Large-Scale Semi-Supervised Dataset for Offensive Language Identification." In: *ArXiv* abs/2004.14454 (2020).

[46] A. Rozental and D. Biton. "Amobee at SemEval-2019 Tasks 5 and 6: Multiple Choice CNN Over Contextual Embedding." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 377–381. DOI: 10.18653/v1/S19-2066. URL: https://www.aclweb.org/anthology/S19-2066.

[47] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: *ArXiv* abs/1910.01108 (2019).

[48] A. Seganti, H. Sobol, I. Orlova, H. Kim, J. Staniszewski, T. Krumholc, and K. Koziel. "NLPR@SRPOL at SemEval-2019 Task 6 and Task 5: Linguistically enhanced deep learning offensive sentence classifier." In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, June 2019, pp. 712–721. DOI: 10.18653/v1/S19-2126. URL: https://www.aclweb.org/anthology/S19-2126.

[49] L. Stappen, F. Brunn, and B. Schuller. "Cross-lingual Zero- and Few-shot Hate Speech Detection Utilising Frozen Transformer Language Models and AXEL." In: *ArXiv* abs/2004.13850 (2020).

[50] P. D. Turney and M. L. Littman. "Measuring Praise and Criticism: Inference of Semantic Orientation from Association." In: *CoRR* cs.CL/0309034 (2003). URL: http://arxiv.org/abs/cs/0309034.

[51] V. N. Vapnik and C. Cortes. "Support-vector networks." In: (1995). URL: https://doi.org/10.1007/BF00994018.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is All you Need." In: *ArXiv* abs/1706.03762 (2017).

[53] S. Wang and Z. Marinho. "Nova-Wang at SemEval-2020 Task 12: OffensEmblert: an Ensemble ofOffensive Language Classifiers." In: *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*. 2020.

[54] Z. Waseem, T. Davidson, D. Warmsley, and I. Weber. "Understanding Abuse: A Typology of Abusive Language Detection Subtasks." In: *CoRR* abs/1705.09899 (2017). arXiv: 1705.09899. URL: http://arxiv.org/abs/1705.09899.

[55] M. Wiegand, M. Siegel, and J. Ruppenhofer. "Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language." In: Sept. 2018.

[56] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. "HuggingFace's Transformers: State-of-the-art Natural Language Processing." In: *ArXiv* abs/1910.03771 (2019).

[57] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore. "Learning from Bullying Traces in Social Media." In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, June 2012, pp. 656–666. URL: https://www.aclweb.org/anthology/N12-1084.

[58] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." In: *NeurIPS*. 2019.

[59] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. "Predicting the Type and Target of Offensive Posts in Social Media." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019, pp. 1415–1420.

[60] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. "SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)." In: *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*. 2019.

[61] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and c. Çöltekin. "SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020)." In: *Proceedings of SemEval*. 2020.

[62] V. Zhou. "Building a Better Profanity Detection Library with scikit-learn." In: (2019). URL: https://victorzhou.com/blog/better-profanity-detection-with-scikit-learn/.

[63] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books." In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 19–27.