A DOMAIN ADAPTATION APPROACH FOR OFFENSIVE LANGUAGE DETECTION WITH BIDIRECTIONAL TRANSFORMERS

by

SUMER SINGH

(Under the Direction of Sheng Li)

ABSTRACT

Offensive language detection (OLD) has received increasing attention due to its societal impact. Recent work shows that bidirectional transformer (BERT) based methods obtain impressive performance on OLD. However, such methods usually rely on large OLD datasets for training. To address the issue of data scarcity in OLD, we propose an effective domain adaptation approach to train bidirectional transformers. Our approach introduces domain adaptation to A Lite BERT (ALBERT), such that it can effectively exploit auxiliary data from source domains to improve the OLD performance in a target domain. Two approaches to domain adaptation are taken. First, we use the auxiliary dataset in an unmodified manner. Next, we modify the auxiliary dataset labels to match the target labels. Experimental results show that the first approach, ALBERT (SA), obtains state-of-the-art performance in most cases. Particularly, our approach significantly benefits underrepresented and underperforming classes, with an improvement of about 40% over ALBERT.

INDEX WORDS: Natural Language Processing, NLP, Transfer Learning, Deep Learning, Preprocessing, Offensive Language Detection, Cyberbullying, Hate Speech, Domain Adaptation

A DOMAIN ADAPTATION APPROACH FOR OFFENSIVE LANGUAGE DETECTION WITH BIDIRECTIONAL TRANSFORMERS

by

SUMER SINGH

B.Tech., Manipal Institute of Technology, INDIA, 2018

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2020

A DOMAIN ADAPTATION APPROACH FOR OFFENSIVE LANGUAGE DETECTION WITH BIDIRECTIONAL TRANSFORMERS

by

SUMER SINGH

ACKNOWLEDGMENTS

I would like to thank my committee members, Dr. Li, Dr. Maier and Dr. Rasheed, for their time, suggestions and inputs throughout my thesis work. A special thanks to my major professor Dr. Li. In my time at University of Georgia, I took two courses with Dr. Li, and completed two projects and my thesis under his guidance. Dr. Li's teaching, advice and guidance has provided me with a solid foundation for my future career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's digital age, the amount of offensive and abusive content found online has reached unprecedented levels. Offensive content online has several detrimental effects on its victims, e.g., victims of cyberbullying are more likely to have lower self-esteem and suicidal thoughts [1]. Offensive content is an umbrella term and many different types of offensive content exist. Subtle differences differentiate types of offensive content, although different types can have extremely different consequences. Thus it is imperative to identify not only if some content is offensive, but also its type.

Some common types of offensive content are:

- **Profanity** Text which includes the use of swear / vulgar words. It is offensive due to the use of a particular word. It's underlying meaning might not be offensive.
- **Threat** A statement of an intention to inflict pain, injury, damage, or other hostile action on someone in retribution for something done or not done.
- **Hate Speech** Abusive or threatening speech or writing that expresses prejudice against a particular group, especially on the basis of race, religion, or sexual orientation.

All three types above are similar in that they are offensive, but an important attribute differentiates them. The target (or victim) of each type is different. Text which is profane could be untargeted, meaning that there is no victim, e.g., 'I am so f-ing happy!' is a profane sentence due to the use of the f-word. It is offensive only due to the presence of a swear word and not because of its underlying meaning. Threats and hate speech are offensive due to their underlying meanings, but a threat is targeted at an individual while hate speech is targeted at a group. Therefore finding the target of offensive content helps us find the type of offensive content. Segregating by type is important because some types of offensive content are more serious and harmful than other types, e.g., hate speech is illegal in many countries and can attract large fines and even prison sentences, while profanity is not that serious.

The aim of this thesis is to work on an automated solution for the task of offensive language detection (OLD). Specifically, not only must the offensiveness of language be detected, but also the target of said language. To this end, the Offensive Language Identification Dataset (OLID) is used, which consists of real-world twitter posts which are labeled using a three-level, hierarchical scheme. The first level, Level A, consists of identifying if a tweet is offensive or not. Next, in Level B, offensive tweets are further categorized as targeted or untargeted. Finally, in Level C, targeted tweets are grouped into targeted at an individual, targeted at a group (race, religion, etc) or targeted at a miscellaneous entity.

The dataset consists of real-world tweets, which signifies that the data is not standardized and substantial preprocessing must take place. Initially, all the data is converted into lowercase and tokenized followed by task specific preprocessing steps. Tweets often contain hashtags, which are words or unspaced phrases, preceded by a hash symbol. These hashtags are converted into meaningful English words, e.g., '#bluesky' is converted into 'blue sky'. Next, several twitter posts contain Emojis, which are small graphics that represent an emotion. All Emojis are converted into their English language representation. Finally, tweets contain several words that are self censored by the poster, e.g., the word 'idiot' might be expressed as 'id*ot'. A mapping is created of common censored forms of words, and using this mapping, censored words are converted into their original uncensored forms.

The dataset is highly imbalanced at each level, i.e., some classes are overrepresented (have a relatively high number of training samples), while others are underrepresented (have a relatively small number of training samples). Training the model without taking this imbalance into consideration could lead to poor performance on the underrepresented classes. This issue is compounded due to the performance metric being macro F1 score, which is calculated by taking the unweighted average of each class's F1 score. This leads to underrepresented classes having an inflated impact on the macro F1 score. Accordingly, a re-weighting scheme is used to alleviate this issue, and consists of prioritizing the underrepresented classes. To this end, each class is assigned a weight which is inversely proportional to its representation. Therefore, during training, a sample from an underrepresented class has higher impact on the model than a sample from an overrepresented class.

The task at hands is a text categorization task and falls into the domain of natural language processing (NLP). To this end, popular and effective NLP models as chosen as baselines. Initially, a support vector machine (SVM) is used on the unpreprocessed data and without class weights, with bag-of-word (BOW) features. Next a convolutional neural network (CNN) based on the architecture of Yoon Kim [2] is used. The CNN is used in three settings: with the unprocessed data and without class weights, referred to as CNN (UP); with the processed data and without class weights, referred to as CNN; and with the processed data and class weights, referred to as CNN (CW). Finally, bidirectional transformers (BERT) [3] and A Lite BERT (ALBERT) [4] are experimented with, using the preprocessed data and class weights.

In the first task, Level A, macro F1 scores of 0.6896, 0.7552, 0.7875, 0.8057, 0.8023 and 0.8109 are achieved by the SVM, CNN (UP), CNN, CNN (CW), BERT and ALBERT respectively. A significant improvement of CNN (CW) over CNN (UP) signifies that the preprocessing steps and re-weighting scheme are effective and required. Comparing the SVM to ALBERT, its observed that most of the improvement is due to ALBERT's superior performance on the underrepresented class, where it beat the SVM's score by 33.3%.

At Level B, macro F1 scores of 0.6288, 0.6732, 0.7038, 0.7348, 0.7433 and 0.7560 are achieved by the SVM, CNN (UP), CNN, CNN (CW), BERT and ALBERT respectively. On analyzing the SVM's per class SVM F1 score, a significant disparity between the overrepresented and underrepresented class is found. Comparing with ALBERT, we note that ALBERT has beat the SVM on the overrepresented class by a margin of only 3.26%, while this margin significantly widens to 66.67% on the underrepresented class.

On Level C, we observe macro F1 scores of 0.4831, 0.4984, 0.5185, 0.5460, 0.5936 and 0.6140 achieved by the SVM, CNN (UP), CNN, CNN (CW), BERT and ALBERT respectively. We observe the F1 score of the CNN is 0 on the samples targeted at miscellaneous entities. The reason for this is twofold: First, the number of samples belonging to this class is extremely low, with only 395 samples. Additionally, this class is a heterogeneous group of samples, making it harder to classify. ALBERT achieves an F1 score of 0.32 on this class, which is an improvement of 137.5% over the SVM. This is similar to the observations made on previous levels, where ALBERT achieves significant gains over the SVM on the underrepresented class.

ALBERT is found to perform the best on each of the three levels, although the performance on the underrepresented classes is found to be significantly worse than on the overrepresented classes. To tackle this issue, a common approach taken is to make use of auxiliary data. Domain adaptation is a transfer learning technique, which consists of using a source domain to provide auxiliary data to improve performance on the target domain, which is the task at hand. In this case the target domain is OLID, and a suitable source domain must be found. The Toxic Comment Classification Challenge (Toxcom) [5] dataset is used as the source domain due to its similarity to OLID. Toxcom consists of Wikipedia comments which are classified into six different types of offensiveness. Two approaches to domain adaptation are applied to the best performing model, ALBERT.

Standard domain adaptation consists of using domain adaptation without modifying the label space in the source domain. The model is first trained in the source domain and its weights from all layers, except the final predictive layer, are saved. The final predictive layer is discarded. Next, the saved model is trained in the target domain with a new predictive layer. Finally, the model is used to predict in the target domain.

Domain adaptation with source label adjustment consists of converting the label space in the source domain to the label space of the target domain. To this end, a label conversion schema is developed to modify the label space in the source domain. After the label space conversion is applied, a model is trained in the source domain and its weights from all layers, including the final predictive layer, are saved. The saved model is then trained in the target domain and finally used to predict in the target domain.

ALBERT is tested with both domain adaptation approaches and different learning rate combinations on the source and target domain are experimented with. ALBERT with standard domain adaptation, ALBERT (SA), achieves macro F1 scores of 0.8241, 0.8108 and 0.6790 on the three levels. The learning rate for these scores is $1.5 * 10^{-5}$ on Toxcom and $2 * 10^{-5}$ on OLID. ALBERT with source label adjustment, ALBERT (LA), achieves scores of 0.8109, 0.7714 and 0.6361 on the three levels. ALBERT (LA) performs better than all approaches without domain adaptation, but is beaten by ALBERT (SA) on all three levels. On analyzing the performance of ALBERT (SA), it is found that the most significant improvements are observed on the underrepresented classes, with improvements of up to 43.75% over ALBERT.

The main contributions of this thesis are:

- Application of a combination of novel and standard preprocessing and feature extraction steps, based on a detailed analysis of the data. First, standard prepocessing steps are applied to normalize and clean the data, followed by dataset specific steps, such as hashtag segmentation, emoji substitution and censored word conversion. Finally, to alleviate the class imbalance issue, a re-weighting scheme is developed and class weights are applied during training.
- Four models are chosen as baselines, namely an SVM, CNN, BERT and ALBERT. Results from the CNN prove that the preprocessing steps and re-weighting scheme are effective. ALBERT is found to perform the best across all three levels and is chosen for domain adaptation.
- Two domain adaptation techniques are developed, mainly to improve performance on the underrepresented classes, and are applied to ALBERT. ALBERT with source label adjustment achieves scores better than all non-domain adaptation approaches. ALBERT with standard domain adaptation achieves the overall best scores, with the scores in Level B and Level C significantly beating current state-of-the-art scores.

This thesis is organized into five chapters. Chapter 1 is Introduction, which aims to summarize the thesis and states the salient details. Chapter 2 is Literature Review, which reviews relevant and related work and introduces the requisite background theory. Chapter 3 is Methodology, in which the preprocessing methods and domain adaptation techniques are described in detail. Chapter 4 is Experiments, which introduces the models used and their settings followed by the experimental results and analysis using the previously described methods. Chapter 5 is Conclusion, which reflects on the results and recommends what work could be done in the future.

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND THEORY

There are many approaches to tackle the issues of offensive content online. The most common technique is to remove offensive contest once it's posted. This is done manually by a group of people, known as moderators or admins. The issue with this method is the sheer volume of posts overwhelm the small number of moderators. Thus an automated technique is required to detect and remove offensive content.

The challenge at hand falls into a fundamental area of NLP, called text classification. Text classification is the process of labeling text according to its content, and in this case the content is a collection of tweets and the labels are the offensiveness and target of tweets. Initial approaches followed rule-based systems, such as looking for keywords, but due to the complexity of language, these rule-based approaches are not effective on a large scale as the number of rules explodes. An alternative to hand-crafted rules is machine learning. Machine learning has rapidly changed over the past decade due to an increase in availability of data and computing resources. It therefore makes sense to broadly split the machine learning techniques used for text classification into three categories.

Classical Machine Learning

Text data poses as inherent problem to machine learning models. As they rely on mathematics, they can not process textual data, which meant that older machine learning text classification techniques relied on a two-step pipeline. First, the textual data must be represented as some meaningful numeric value, and this step is known as feature extraction. Once the features are extracted, they are passed into a machine learning model.

- **Feature Extraction Methods.** The simplest feature extraction model would be bag-of-words (BOW) models. BOW uses counts of words occurring as their features, i.e., each text sample is converted into a vector in which each element corresponds to the count of a particular word in that text sample. Each text sample is converted to a vector of a fixed dimension, which corresponds to vocabulary size and out of vocabulary words are ignored. An extension to BOW would be n-grams, which uses the counts of n-length sequences. Thus, an unmodified BOW representation could be called 1-gram (or unigram), while a 2-gram (or bigram) model uses the counts of 2-length sequences, and so on. For example, the sentence 'How are you today?' is broken down into 'how are', 'are you' and 'you today' for a bigram model.

  BOW are useful because they are simple to use and understand. The drawback is that the vectors are based only on counts of words. To fully understand textual content, one must not only look at the counts of words, but also their order, context and relation to other words. All this important information is disregarded in this technique and, e.g., the sentence 'I am happy, not sad' and 'I am sad, not happy' are interpreted as exactly the same.

  Additionally, domain specific hand-crafted features can be created. These techniques are tied to the task at hand, which means they do not generalize well and might require expert domain knowledge.

- **Models.** Commonly used classical machine learning models include Naive Bayes and SVMs. These methods are highly effective when less data is available and they require modest computing resources.

Deep Learning

Deep learning is a subset of machine learning which is based on the neural network. Neural networks consist of layers of units, with each layer connected to the next. Each successive layer is able to learn a more complex representation of the previous layer's data. Deep learning uses this concept to extract higher-level (more complex) features in a layer wise fashion. The word *deep* is related to the number of layers in a neural network. Generally, a network is *deeper* than another network if it has more layers.

Similar to older machine learning models, deep learning models require a way to convert text data into meaningful numeric data. Commonly used methods are to use pretrained word embeddings, such as word2vec [6] or glove [7], which map each word to a predefined high dimensional numeric vector. Word embeddings contain more information than BOW because each word's vector representation is based on its actual meaning, not just its count. The biggest drawback to this method is that each particular word has only one embedding. That means polysemy (the coexistence of many possible meanings for a word or phrase) is inadequately represented, e.g., the word 'bank' in the sentence 'The bank was robbed.' and 'I sat on the river bank.' will have the same vector representation in spite of having different meanings. There are various types of deep learning model used for text classification, differentiated by their architecture:

- **Recurrent Neural Networks (RNNs).** RNNs view the input data, usually text, as a sequence. This means that not only the word itself, but also its relative position is taken into account and as such, RNNs are a natural fit for language tasks. Due to the sequential nature of the input, many long term dependencies are introduced which can lead to the vanishing or exploding gradient problem. LSTMs [8] and GRUs [9] are modified version of RNNs which significantly reduce the issues caused by the long term dependencies. Nevertheless, RNNs still suffer from lack of parallelization due to their sequential nature, which makes them slow to train.
- **Convolutional Neural Networks (CNNs).** CNNs are known to be highly effective on image based tasks as CNNs work well in detecting patterns. Patterns do occur in language, hence people have been applying CNNs to language tasks as well. CNNs are normally used at a character level (where each input is a character) or at a word level (where each input is a word). Word level CNNs use word embeddings, such as glove or word2vec, to convert the input data.
- **Hybrid Networks.** Hybrid network use a combination of different architectures to improve performance. For example, CNNs and RNNs might be used in conjunction.

Transformer Networks

Transformers [10] were developed to solve the issue of lack of parallelization faced by RNNs. Transformers calculate a score for each word with respect to every other word, in a parallel fashion. The score between two words signifies how related they are. Due to the parallelization, transformers train rapidly on modern day GPUs. Current day state of the art techniques, such as BERT [3], XL-NET [11] and ALBERT [4] are all based on the transformer architecture.

- **BERT.** BERT is a transformer based network architecture, mainly used for NLP tasks. Using BERT for a task consists of two steps:

  1. **Pretraining.** It consists of training the model is an unsupervised fashion on huge amounts of unlabeled data. BERT uses two tasks for pretraining, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM consists of masking input words and training the model to correctly predict the masked word. NSP consists of a two sentence input, and training the model to correctly predict if one sentence follows the other. After pretraining is complete, the model weights are saved and passed to the next step.
  2. **Finetuning.** It consists of additionally training the pretrained BERT network on a downstream language task. Finetuning occurs in a supervised fashion and is task specific.

  Pretraining is task independent and must be done before using BERT for any language task. Pretrained BERT models are available on the internet and pretraining can be skipped by using a pretrained model. Our experiments are only concerned with finetuning.

- **ALBERT.** ALBERT is modification of BERT, and focuses on reducing BERT's memory requirement and increasing its training speed. ALBERT does this using two techniques:

  1. **Factorized embedding parameterization.** ALBERT factorizes the embedding parameters and decomposes them into two smaller matrices. This unties the embedding layer size from the hidden layer size, resulting in a reduction of parameters.
  2. **Cross-layer parameter sharing.** ALBERT shares parameters across all layers, which results in a smaller number of parameters.

Domain Adaptation is a transfer learning technique used primarily in deep learning. It aims to apply knowledge learnt in a particular task on another related task. It consists of using a well performing model on a source dataset. In this paper, two domain adaptation approaches are taken, which we call **standard domain adaptation** and **domain adaptation with source label adjustment**.

2.2 RELATED WORK

Classical machine learning approaches are a good starting point for an NLP task, and can be effective in the appropriate circumstances. Modifying them with additional hand-written rules can, in some situations, outperform more complex models. Radivchev and Nikolov [12] work on OLID with multiple models, including an SVM, with which they achieve their best score of 0.6674 on Level B. Han et al. [13] work on OLID, and develop a Modified Sentence Offensiveness Calculation (MSOC) model, which uses a dictionary of offensive words to calculate an offensive score for a sample. On level B, their MSOC model significantly outperforms their RNN based model, proving its effectiveness. The downside to using such models is the requirement of domain knowledge and that it might fail to scale to larger and more complex tasks. Davidson, et al. [14] worked on the task of detecting offensive language and hate speech using an SVM, with which they achieve an F1 score of 90%. Their task differs from OLID as they do not differentiate between tweets targeted at a group and individual, which is a key requirement in OLID. Chavan and Shylaja [15] work on automatically detecting instances of cyber-aggression, and report an AUCROC score of 86.92% using an SVM with skip-grams. Including skip-gram features gives them an increase of around 9% over their previous best score, but their task does not consist of further categorizing offensive samples based on the target.

Deep learning models outperform classical machine learning models in most cases, provided there is enough training data. Liu et al. [16] experimented with LSTMs on OLID, and surprisingly, their logistic regression model outperforms their LSTM on two out of three tasks. In this work, the CNN outperforms the SVM in all three tasks, which suggests that CNNs might be more appropriate than RNNs on OLID. Han et al. [13] use a GRU model on OLID, but it is outperformed on two out of three tasks by the previously described MSOC model. We note that RNN based models, such as GRUs and LSTMs, are reported to have relatively lower performance on OLID. This could be due to the long term dependencies present in the tweets, and prompts us to instead use a CNN based model. Stammbach et al. [17] work on a dataset consisting of German tweets, with 5009 training samples. For prepossessing, they remove all non-alphanumeric characters, including emojis, and strip the '#' symbols before hashtags, but do not split the hashtag itself. They experiment with two models, one CNN based and one RNN based. They experimented with GRUs and LSTMs and found that GRUs performed better than LSTMs. They report a best F1 score of 78.6% using their CNN model. Once again, we observe CNN based models outperforming RNNs. Georgakopoulos et al. [18], work on the Toxcom[5] dataset, in which the task is to classify Wikipedia comments into 6 categories, namely 'toxic', 'severe toxic', 'obscene', 'threat', 'insult', and 'identity hate'. Each comment can belong to 0 or more categories, making it a multi label dataset. The authors use several models, including a CNN based network [2] with word2vec, SVMs, naive bayes and linear discriminant analysis. The authors find their CNN based model (accuracy of 0.912) significantly outperform BOWs based models (highest accuracy of 0.811). All the above works illustrate the effective of CNNs for OLD tasks, and is thus chosen by us, over RNNs.

Currently, the leading models across a wide range of NLP task are transformer based models, such as BERT [3], XL-NET [11] and ALBERT [4]. Liu et al. [16] acheive their best score on all three tasks on OLID using BERT, which outperforms the logistic regression and LSTM model. Radivchev and Nikolov [12] achieve their best score on two out of three tasks on OLID with BERT. The main limitation with BERT is its large memory footprint, which restricts most users to its base model. ALBERT, a modification to BERT, was introduced with the main aim of reducing the memory requirement of BERT. On complex NLP tasks, such as the Squad dataset [19], which is a question-answering dataset, ALBERT based models currently hold four out of five of the top scores. With this in mind, BERT and ALBERT are chosen as our final models.

Deep learning networks and transformer based networks are able to efficiently process high amounts of data, which indicates that using these networks with small amounts of training data is sub-optimal. In many cases, the data is limited and it is difficult or impossible to collect more data, but in most NLP tasks, we can reuse data from different tasks. This is possible due to the similarity in data across different NLP tasks, e.g., the Toxcom dataset and OLID both consist of classifying user generated content (Wikipedia comments and Tweets respectively), based on their offensiveness. Thus we can supplement data from one task with data from another similar task, which eliminates the need to collect more data. An effective way to achieve this is with the application of domain adaptation. In this work, two domain adaptation approaches are developed, and used to improve the performance of best performing network, ALBERT.

CHAPTER 3

METHODOLOGY

This chapter consists of formally defining the problem at hand and the techniques undertaken to solve the challenge. The first section outlines the problem and the motivation and need to work on it. The second section describes the dataset used in the problem, followed by an analysis of the dataset and the preprocessing steps taken to improve the quality of the data. The third section introduces domain adaptation, which is an approach taken to improve the performance of a network on a particular task by training it on a different, but similar, task.

3.1 PROBLEM DEFINITION

The rise of social media has led to a sharp increase in the amount of offensive content generated online, which in turn has severe detrimental effects on society. The need for an efficient and automated tool to detect and categorize offensive content is the main motivation for this work. The task an hand is an offensive language detection (OLD) problem, which consists of detecting if some textual content is offensive, i.e., abusive, threatening, prejudiced, racist, etc, followed by grouping the offensive content based on its target. Accordingly, the dataset chosen is the Offensive Language Identification Dataset (OLID) and consists of detecting if tweets are offensive or not, followed by further categorizing offensive tweets based on if they were targeted at something or untargeted, and finally categorizing the targeted tweets based on who or what they were targeted at.

Four models are experimented with as baselines, namely an SVM, CNN, BERT and ALBERT. Domain adaptation is applied to the best performing model to further improve it. Domain adaptation consists of training a model in one or more source domains, followed by training and predicting in the target domain. The source domain is a domain related and similar to the target domain, which can be used to improve model performance in the target domain. The target domain is the task at hand, in which a performance boost is required, and in this case, OLID is the target domain.

3.2 DATASET

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 1: Dataset details across all three levels. The dataset is highly imbalanced at each level.

The dataset used is the Offensive Language Identification Dataset (OLID) [20]. The dataset consists of real world tweets. This dataset is unique in that it consists of a three level hierarchical structure to classify tweets. The three levels are:

- Level A - Offensive Language Detection. Level A is the most general level and categorizes tweets into two categories:
  - Not Offensive *(NOT)*: Posts that are unoffensive in all aspects.
  - Offensive (*OFF*): Tweets that are offensive in any form. This includes threats, hate speech, profanity, etc.
- Level B - Categorization of Offensive Language. Level B further categorizes Offensive (*OFF*) tweets from Level A into:
  - Targeted Insult (*TIN*): Tweets that are targeted at someone or something, e.g., threats, hate speech, etc.
  - Untargeted (*UNT*): Tweets that are offensive but untargeted, e.g., obscene language.
- Level C - Offensive Language Target Identification. Level C further categorizes Targeted Insults (*TIN*) tweets from Level B into:
  - Individual (*IND*): Offensive tweets that are targeted at an individual, such as insults or threats.

- Group (*GRP*): Tweets that are targeted at a particular group, e.g., religion, race, etc. These tweets are often called hate speech.
- Other (*OTH*): Tweets that are targeted at miscellaneous entities, e.g., an event, organization, etc.

Dataset Analysis

Tweet	A	B	C
@USER He is so generous with his offers.	NOT	-	-
IM FREEEEE!!!! WORST EXPERIENCE OF MY FUCKING LIFE	OFF	UNT	-
@USER Principled conservatives are #Hypocrickets	NOT	-	-
@USER Holder needs to be prosecuted	NOT	-	-
@USER Fuk this fat cock sucker	OFF	TIN	-

Table 2: Sample tweets from the dataset. Note the usage of '@USER' tags and hashtags.

Looking at Table 2, several user tags in the form of '@USER' are present. This is because twitter posts often contain tags which connect them to other users. To protect the privacy of users, all the tags are anonymized to '@USER'. This suggests that these tags do not contain much information and should be removed. Twitter users often use hashtags, as shown in the third tweet ('#Hypocrickets'). These hashtags contain useful information but can not be used in their current form. We also notice the use of several Emojis across the datasets. Emojis, too, contain useful information and can help us classify the tweet.

Preprocessing

The dataset consists of real-world data in the form of tweets. This means the data is not standardized and significant preprocessing is required to clean the data. Uncleaned data contains lots of noise and can lead to much lower performance. Initially, all data is converted to lowercase and tokenized. '@USER' tokens are removed. Task specific preprocessing steps are described below.

Hashtag Segmentation

Hashtags are commonly used on Social Networking sites such as Twitter. Hashtags are used in tweets to connect a tweet to a specific topic. The format of a hashtag is a hash symbol in front of a word or unspaced phrase in a message. For example the hashtag '#bluesky' can be used in tweets with pictures of the sky.

Hashtags contain important metadata related to the tweet and can help the model classify the tweet, but hashtags in their raw form pose an issue to neural networks. Current NLP networks work by mapping known words to vectors. The list of known words is finite and includes commonly used English words. Hashtags are not English words, due to the preceding hash symbol and the convention of not using spaces between words. To help the model make sense of hashtags two things must be done. First, the hash symbol is removed and this step is trivial. Next, the word is split up and this second step is done using wordsegment[1]. The hashtag will be converted to plain English after these two steps, e.g., '#bluesky' will become 'blue sky'.

Censored Word Conversion

Offensive tweets often contain vulgar words. These vulgar words are informative to the model and help classify the tweet. Oftentimes people self-censor offensive words used in their tweet and though this is polite, it is harmful to our model, e.g., people might censor the word 'sample' as 's@mple', 's!mple', 'sa#ple', etc. A human is able to interpret all these forms correctly as 'sample', but a machine learning model will consider them to be different tokens. To alleviate this issue, a mapping is created of offensive words and their commonly used censored forms. All the censored forms are converted to their uncensored forms.

Emoji Substitution

Emojis are graphical symbols used in electronic text mediums, such as tweets, to convey additional meaning. Commonly used Emojis include Smiley Face, Thumbs Up, Sad Face, etc. Computers

[1]`https://github.com/grantjenks/python-wordsegment`

read Emojis as Unicode and hence they have no meaningful language representation. Therefore, using Emojize[2], all emojis are converted to their English representation and this allows them to be passed into the network as if they were English words.

Class Weights

Level	A		B		C		
Classes	NOT	OFF	TIN	UNT	IND	GRP	IND
Samples	8,840	4,400	3,876	524	2,407	1,074	395
Class Weight	1.00	2.009	1.00	7.394	1	2.517	6.845

Table 3: Class weight details. Underrepresented classes are assigned higher weights and vice versa.

The dataset is highly skewed at each level and assigning equal weights to every class might cause low performance on the minority classes. To alleviate this issue, assigning different class weights can be used. Underrepresented classes will be assigned a higher weight, and vice versa. These weights are used during training and ensure that the model gets a higher reward/penalty for correctly/incorrectly classifying a minority class and vice versa. Formally, let the classes be $\{c_1, c_2, \cdots, c_k\}$ and number of samples in each class be $\{N_1, N_2, \cdots, N_k\}$, then class c_i is assigned a weight proportional to N_i^{-1}. A particular class is given the inverse of its representation as a class weight. For example, if there are two classes, A and B with 10 and 20 samples respectively. Class A will be assigned the weight 2 and class B will be assigned the weight 1.

Max Sequence Length and Long Sentences

BERT and ALBERT have a hyperparameter called max sequence length and if a sequence is longer than max sequence length, the extra words will be ignored. Longer max sequence lengths are better because they will capture more information, but the downside is that more memory is required. In the case of this task, it is important to think of sentences that are longer than the max sequence length. Consider a sentence with 20 words labeled as offensive and a max sequence length of 18. That means the last two words of the sentence will be ignored by the model. Consider the

[2] `https://github.com/carpedm20/emoji`

case in which this sentence was offensive only because of the last two words. The model will be trained to consider the new shortened sentence offensive, in spite of it containing nothing offensive. Due to this, we discard all training samples which are longer than the max sequence length.

Source Domain

Classification	**# of instances**
clean	143,346
toxic	15,294
obscene	8,449
insult	7,877
identity hate	1,405
severe toxic	1,595
threat	478

Table 4: Toxcom training dataset statistics. The *clean* class has significantly more training samples than all the other classes.

The Toxic Comment Classification Challenge (Toxcom) [5] dataset is used as a source domain for the purpose of domain adaptation. It is similar to OLID, with some key differences. The Toxcom dataset consists of Wikipedia comments which have been labeled by human raters for toxic behavior. It is a non-hierarchical, multi-label dataset, i.e., samples can belong to zero or more classes. It consists of 6 classes. We introduce a new class called *clean.* Samples that belong to none of the 6 classes are labeled as *clean.* Details given in Table 4.

3.3 DOMAIN ADAPTATION

OLID is small and few classes are significantly underrepresented, with the *OTH* class having just 395 training samples. We can use a related source domain (dataset) to provide more training data. The data in this domain must be similar to the target domain. Data from multiple source domains can be used, but we use only one. The source dataset is initially used to train the model. The same model is then trained using the target data. In this case, OLID is the target domain and Toxcom is the source domain. Two domain adaptation approaches are used, which we call **standard domain adaptation** and **domain adaptation with source label adjustment**.

Standard Domain Adaptation

Standard domain adaptation (SA) is applying domain adaptation, without modifying the labels in the source domain. As the label space in each domain differs, a new predictive layer, specific to each domain, must be used. Different learning rate combinations can be used for training in the source and target domain. The steps in this approach are:

1. Train the network on the source domain (lines 6 - 11 in Algorithm 1).
2. Freeze all the layers and discard the final predictive layer.
3. Reuse the previously frozen layers with a new predictive layer and train the network on the target domain (lines 12 - 20 in Algorithm 1). There are several ways to treat the previously frozen layers in this step:
    (a) A feature extraction type approach in which all layers remain frozen, i.e., not trained through backpropagation. This corresponds to $K = \{0, 1, \cdots, L-2\}$ in Algorithm 1.
    (b) A finetuning type approach in which all layers are finetuned, i.e., trained through backpropagation. This corresponds to $K = \{\emptyset\}$ in Algorithm 1.
    (c) A combination of both in which some layers are finetuned while some are frozen.
4. Predict in target domain.

Algorithm 1: Standard domain adaptation training algorithm. L is the number of layers, including the predictive layer, n is the number of training samples in the source domain, m is the number of training samples in the target domain, and K is the set of layers that remain frozen during training in the target domain.

```
1  Initialize;
2  weight vectors $\{w_0, w_1, \cdots, w_{L-1}\}$;
3  Source domain samples $\{s_0, s_1, \cdots, s_{n-1}\}$;
4  Target domain samples $\{t_0, t_1, \cdots, t_{m-1}\}$;
5  K = $\{x \in \mathbb{N} | 0 \leq x < L-1\}$;
6  for $i \leftarrow 0$ to $n$ do
7      forward propagation on $\{s_i\}$;
8      for $j \leftarrow 0$ to $L-1$ do
9          gradient descent on $\{w_i\}$
10     end
11 end
12 Initialize;
13 weight vector $\{w_{L-1}\}$;
14 for $i \leftarrow 0$ to $m$ do
15     forward propagation on $\{t_i\}$;
16     for $j \leftarrow 0$ to $L-1$ do
17         if $j \notin K$ then
18             gradient descent on $\{w_i\}$
19     end
20 end
```

Domain Adaptation with Source Label Adjustment

Toxcom Labels						OLID Labels		
Toxic	Severely Toxic	Obscene	Insult	Threat	identity hate	Level A	Level B	Level C
0	0	0	0	0	0	NOT	-	-
1	x	x	0	0	0	OFF	UNT	-
x	1	x	0	0	0	OFF	UNT	-
x	x	1	0	0	0	OFF	UNT	-
x	x	x	1	x	0	OFF	TIN	IND
x	x	x	x	1	0	OFF	TIN	IND
x	x	x	x	x	1	OFF	TIN	GRP

Table 5: Toxcom conversion scheme to OLID. x means the label could be 0 or 1. The scheme is developed using the English language meaning of Toxcom labels.

Domain adaptation with source label adjustment (LA) is using domain adaptation, but modifying the source domain labels to be in the same space as the target domain label, i.e., source and target datasets must be labeled in the same format. This allows the model to share the weights of all layers, including the predictive layer.

The steps in this approach are:

1. Modify the source domain labels to be in the same format as the target domain (line 6 in Algorithm 2).
2. Train the network in the source domain (line 7 - 12 in Algorithm 2).
3. Freeze all the layers including the predictive layer.
4. Reuse the previously frozen layers, including the predictive layer, and train in the target domain (line 13 - 18 in Algorithm 2).
5. Predict in the target domain.

To use this method on our datasets, the Toxcom dataset must be modified to be in the same format as OLID. To do this, a conversion scheme is created manually, using the English language interpretations of the labels. For example, a threat is an offensive comment, targeted at an individual. Therefore a comment labeled as *threat* in the Toxcom dataset should be labeled as *OFF*, *TIN* and *IND* on Level A, B and C respectively in OLID. Using this logic, a conversion scheme is developed to convert Toxcom samples into OLID format. The conversion schema used is outlined in Table 5.

Algorithm 2: Domain adaptation with source label adjustment training algorithm. L is the number of layers, including the predictive layer, n is the number of training samples in the source domain and m is the number of training samples in the target domain.

```
1  Initialize;
2  weight vectors $\{w_0, w_1, \cdots, w_{L-1}\}$;
3  Source domain samples $\{s_0, s_1, \cdots, s_{n-1}\}$;
4  Target domain samples $\{t_0, t_1, \cdots, t_{m-1}\}$;
5  K $= \{x \in \mathbb{N} | 0 \leq x < L - 1\}$;
6  Convert source domain labels using conversion scheme;
7  for $i \leftarrow 0$ to $n$ do
8      forward propagation on $\{s_i\}$;
9      for $j \leftarrow 0$ to $L - 1$ do
10         gradient descent on $\{w_i\}$
11     end
12 end
13 for $i \leftarrow 0$ to $m$ do
14     forward propagation on $\{t_i\}$;
15     for $j \leftarrow 0$ to $L - 1$ do
16         gradient descent on $\{w_i\}$
17     end
18 end
```

For both domain adaptation approaches, we make use of variable learning rates. This means that the learning rate while training in the source domain could be different from the learning rate in the target domain.

CHAPTER 4

EXPERIMENTS

This chapter consists of the details of the experiments undertaken and their results. The first section describes the experimental settings, which includes models used, parameters, evaluation metric, etc. The next three sections describe and analyze the results of Level A, B and C respectively. The final section illustrates and analyzes the domain adaptation approach, concluding with a comparison of the baselines with the standard domain adaptation approach.

4.1 EXPERIMENTAL SETTINGS

In the experiments, four representative models are used as baselines, including the support vector machine (SVM), convolutional neural networks (CNN), BERT and ALBERT. The CNN is implemented based on the architecture by Yoon Kim [2], and consists of two channels. Both channels are initialized with glove embeddings, but only one channel is trained through backpropagation, while the other channel remains static. We evaluate CNN in three different settings, as outlined in Table 6. We use the base version of BERT and the large version of ALBERT. The max sequence length is set to 32 and 64 for BERT and ALBERT, respectively. Training samples with length longer than max sequence length are discarded. Moreover, we compare our approach with three state-of-the-art methods [16, 13, 12] on OLID. The metric used is macro F1 score. Macro F1 score is calculated by taking the unweighted average of each class's F1 score. Method details are found in Table 6 and results are found in Table 7.

4.2 LEVEL A

The SVM achieves a macro F1 score of 0.6896. Looking at Figure 1, we observe a significant disparity between the per class F1 scores of 0.83 vs 0.54. The SVM is struggling to classify offensive tweets.

Model	Preprocessing	Class Weights	Domain Adaptation
SVN	No	No	No
CNN (UP)	No	No	No
CNN	Yes	No	No
CNN (CW)	Yes	Yes	No
BERT	Yes	Yes	No
ALBERT	Yes	Yes	No
ALBERT (SA)	Yes	Yes	Standard Domain Adaptation
ALBERT (LA)	Yes	Yes	Domain Adaptation with Source Label Adjustment

Table 6: Model details. Domain adaptation is applied to ALBERT as it performs the best.

Model	Level A	Level B	Level C
NULI [16]	**0.8286**	0.7159	0.5598
JHAN [13]	0.6899	0.7545	0.5149
Nikolov-Radivchev [12]	0.8153	0.6674	0.6597
SVM	0.6896	0.6288	0.4831
CNN (UP)	0.7552	0.6732	0.4984
CNN	0.7875	0.7038	0.5185
CNN (CW)	0.8057	0.7348	0.5460
Bert	0.8023	0.7433	0.5936
Albert	0.8109	0.7560	0.6140
Albert (LA)	0.8191	0.7714	0.6361
Albert (SA)	0.8241	**0.8108**	**0.6790**

Table 7: Results. First three rows are previous state of the art results at each level. State of the art results are achieved by ALBERT (SA) in Level B and Level C.

Train			Test		
Word	Count	Ratio	Word	Count	Ratio
bitch	100	0.98	shit	24	0.88
idiot	55	0.96	can	32	0.63
coward	21	0.95	liberals	81	0.52
asshole	19	0.95	than	22	0.50
bitches	15	0.93	their	35	0.49
fuck	180	0.92	should	25	0.48
stupid	121	0.92	more	44	0.48
morons	12	0.92	get	51	0.47
stupidity	12	0.92	they	78	0.46
deals	11	0.91	think	22	0.45

Table 8: Top 10 strongest words associated with offensive tweets. The offensive tweets from the training set are commonly associated with swear words, while this is not the case in the test set.

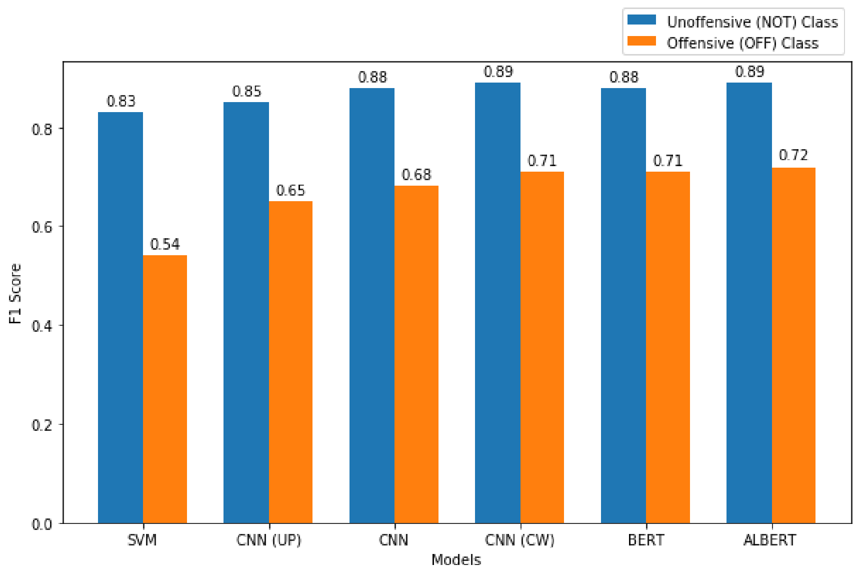

Figure 1: Classwise F1 scores for Level A. The performance on the majority class, *NOT*, does not improve significantly across models.

To analyze the poor performance on offensive tweets, we find the words highly associated with them, for the training and test sets. To get an association ratio, we calculate the number of times the word is seen in an offensive sample divided by the total number of times the word appears. On the training set, Table 8, we observe that the words mostly associated with offensive tweets are swear words. This signifies that the training set consists of several offensive tweets which include profanity. Conversely on the test set, the strongly associated words do not consist of many swear words, meaning the offensive tweets in the test set do not often contain profanity. The SVM has therefore strongly associated non-profane tweets to be non-offensive, which is evidently not the case.

CNN is used is three different settings, outlined in Table 6. Glove embeddings are used for word embeddings. Initially, using the raw data, it gives a score of 0.7552. This consists of an F1 score of 0.85 and 0.65 on *NOT* and *OFF* samples respectively. To increase the performance we make use of data preprocessing, outlined in chapter 3.2.

The new data gives a score of 0.7875 with an F1 score of 0.88 and 0.68 on *NOT* and *OFF* samples respectively. A significant improvement in performance is noted, proving that the preprocessing steps were effective. Although the score increased, the disparity in performance remains between the two classes, which can be attributed to class imbalance

To alleviate the imbalanced class issue, we make use of class weights, described in chapter 3.2. Training with class weights gives us an improved score of 0.8057. This consists of an F1 score of 0.89 and 0.71 on *NOT* and *OFF* samples respectively. Using class weights boosts the overall performance and is mainly due to an improvement of the F1 score of *OFF* samples from 0.68 to 0.71. Overall, an improvement of 17 points or 33% is observed from the SVM's F1 score of 0.54 on *OFF* samples.

BERT is used to further improve performance on this dataset. BERT achieves an F1 score of 0.8023, which is lower than the CNN's score. This can be attributed to the fact that BERT's base version is used, not the large one and this was done due to memory constraints. ALBERT achieves a macro F1 score of 0.8109, which is the best score achieved till now and significantly better than SVM's score of 0.6896. Majority of this improvement can be attributed to ALBERT's superior performance on the underrepresented *OFF* samples, where it has improved upon the SVM's score of 0.54 to 0.72 or 33.3%.

4.3 LEVEL B

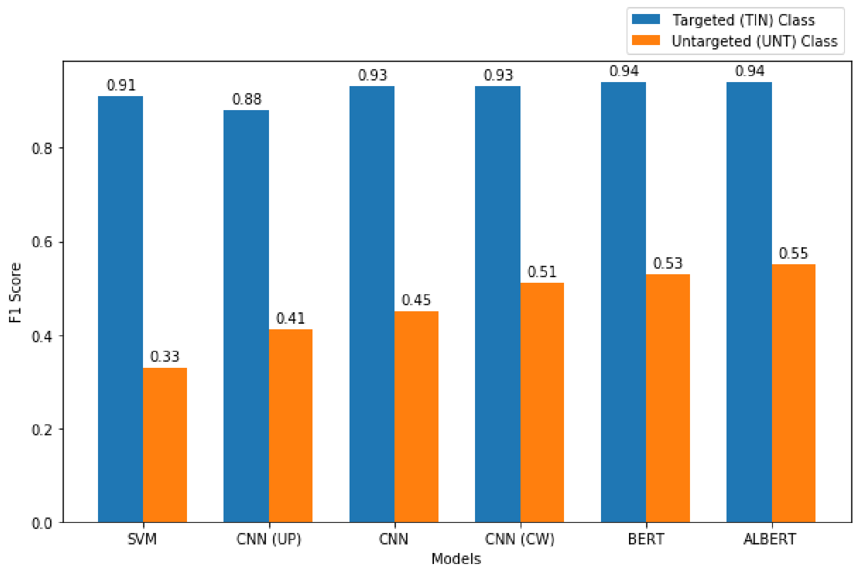

Figure 2: Classwise F1 scores for Level B. All models perform poorly on the underrepresented class, *UNT*, which can be attributed to its low number of training samples.

SVM achieves a macro F1 score of 0.6288 on task B and it consists of an F1 score of 0.91 on *TIN* samples and 0.33 on *UNT* samples. The inter-class score disparity is larger than Level A. This can be attributed to the significant imbalance of training data. There are just 524 *UNT* samples versus 3,876 *TIN* samples.

The CNN gives a score of 0.6732 on the unprocessed data. This is made up of an F1 score of 0.88 on *TIN* samples and 0.41 on *UNT* samples. Preprocessing the data improves the overall score to 0.7038. Making use of class weights significantly boosts the overall performance to 0.7348, which consists of an F1 score of 0.93 and 0.51 on *TIN* and *UNT* samples respectively. We can observe the effect of preprocessing and using class weights on the underrepresented class, as the F1 score of *UNT* samples jumped for 0.41 to 0.51, which is an increase of 24%.

ALBERT achieves a score of 0.7560 on Level B. It consists of 0.94 on *TIN* samples and 0.55 on *UNT* samples. Compared to the SVM, ALBERT has improved only 0.03 points or 3.26% on *TIN* samples. Conversely, it has significantly better performance of 0.22 points or 66.67% on *UNT* samples. This is similar to the results on Level A, where we see the majority of improvement from SVM to ALBERT is due to ALBERT's performance on the minority class.

We notice a trend developing. ALBERT shines when training data is sparse. This can be attributed to the intensive pretraining ALBERT goes through. This allows ALBERT to preemptively have a firm grasp of the language, enabling it to perform well even in the case of sparse finetuning data.

4.4 LEVEL C

SVM achieves a macro F1 score of 0.4831 on task C, which consists of an F1 score of 0.69, 0.59 and 0.16 on *IND*, *GRP* and *OTH* samples respectively. There are only 395 *OTH* samples in the training dataset, which is arguably leading to poor performance on those samples.

The CNN gives a score of 0.4984 on the unprocessed data. This consists of an F1 score of 0.78, 0.7 and 0.0 on *IND*, *GRP* and *OTH* samples respectively. The lack of training data, preprocessing and appropriate class weights has resulted in the CNN failing to correctly classify even 1 *OTH*

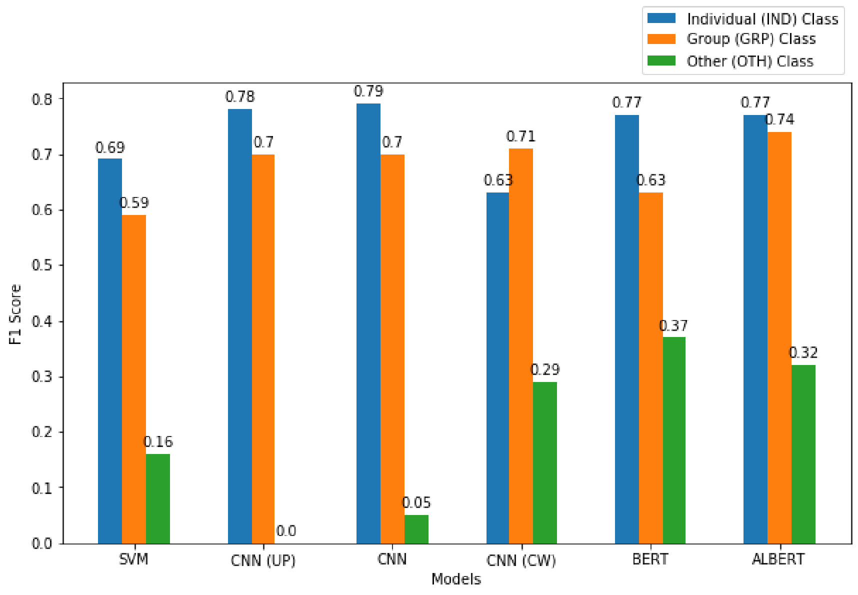

Figure 3: Classwise F1 scores for Level C. Performance on the *OTH* class is poor for all models, and this can be attributed to it being a vague class with a low number of training samples.

sample.

Making use of preprocessing increases the macro F1 score to 0.5185, with the F1 score of *OTH* samples rising to 0.05. Adding class weights significantly improves performance to 0.5460, with F1 scores of 0.63, 0.71 and 0.29 on *IND*, *GRP* and *OTH* samples respectively. Performance has decreased slightly on *IND* and *GRP* samples, but significantly improved to 0.29 on *OTH* samples.

BERT and ALBERT achieve a macro F1 score of 0.5936 and 0.6140 respectively. BERT has achieved the best F1 score on the underrepresented *OTH* samples of 0.37. This is an improvement of 0.17 points or 131.5% over the SVM. This is in line with the trend observed. BERT and ALBERT are able to perform relatively much better than other models on the underrepresented classes due to their pretraining.

4.5 DOMAIN ADAPTATION

A constant and obvious trend is observed across all tasks and models: The classes with fewer training samples are significantly harder to classify. ALBERT significantly outperforms all other networks and is thus chosen for domain adaptation. Two domain adaptation approaches are followed, as described in chapter 3.3.

Standard Domain Adaptation

Toxcom Learning Rate	OLID Learning Rate	Level A	Level B	Level C
2.00 * 10^{-5}	2 * 10^{-5}	0.8082	0.7544	0.6170
1.00 * 10^{-5}	2 * 10^{-5}	0.8076	0.7759	0.6113
3.00 * 10^{-5}	3 * 10^{-5}	0.8071	0.7254	0.6041
2.00 * 10^{-5}	3 * 10^{-5}	0.8055	0.7320	0.6087
1.00 * 10^{-5}	1.5 * 10^{-5}	0.8005	0.7627	0.6208
1.5 * 10^{-5}	2 * 10^{-5}	**0.8241**	**0.8108**	**0.6790**

Table 9: Standard domain adaptation results. Learning rate in the target domain is always set to be higher than or equal to learning rate in the source domain.

Many different learning rate combinations are experimented with. The learning rate for the target dataset is always set to be higher or equal to the learning rate of the source dataset. This is because more weightage must be given to the training samples of the target set. A learning rate of between 1 * 10^{-5} and 2 * 10^{-5} seems to be optimal. A learning rate of 3 * 10^{-5} on both datasets gives the worst results on Level B and C. The best results are achieved with a learning rate of 1.5 * 10^{-5} on Toxcom and 2 * 10^{-5} on OLID.

Domain Adaptation with Source Label Adjustment

New Samples from Toxcom	Level A	Level B	Level C
143,346	NOT	-	-
7,940	OFF	UNT	-
6,880	OFF	TIN	IND
1,405	OFF	TIN	GRP

Table 10: Number of new samples from Toxcom after it is converted into OLID format. No new samples from the *OTH* class are created, as it was not possible to define a conversion rule for this class.

Toxcom dataset is modified, as described in Table 5, to be in the same format as OLID. The details of the modified dataset are given in Table 10.

Toxcom Learning Rate	OLID Learning Rate	Level A	Level B	Level C
2.00 * 10^{-5}	2 * 10^{-5}	0.8021	0.7450	0.6270
1.00 * 10^{-5}	2 * 10^{-5}	0.8084	0.7355	0.6245
3.00 * 10^{-5}	3 * 10^{-5}	0.8082	0.7196	0.6102
2.00 * 10^{-5}	3 * 10^{-5}	0.8094	0.7227	0.6099
1.00 * 10^{-5}	1.5 * 10^{-5}	0.8089	0.7254	0.6208
1.5 * 10^{-5}	2 * 10^{-5}	0.8053	0.7407	0.6167
0.5 * 10^{-5}	2 * 10^{-5}	0.8040	0.7453	0.6142
0.25 * 10^{-5}	2 * 10^{-5}	**0.8109**	**0.7714**	**0.6361**
0.125 * 10^{-5}	2 * 10^{-5}	0.8021	0.7320	0.6126

Table 11: Domain adaptation with source label adjustment results. Results are worse than the standard domain adaptation approach.

143,346 new *NOT* samples and 16,225 *OFF* samples and created using the conversion schema. To avoid a bias towards *NOT* samples, we only consider 16,225 randomly sampled *NOT* comments. There are no new samples that belong to the Level C class *OTH*. This is because the *OTH* class consists of miscellaneous samples and is thus a vague category. Therefore creating a conversion rule for this class is not possible.

The experiments, details in Table 11, start off with the same learning rate combinations as the previous method. Higher learning rates on Toxcom result in significant degradation in performance. Conversely, lower learning rates on Toxcom result in an improvement of performance, with the best results achieved at a learning rate of 0.25 * 10^{-5} and 2 * 10^{-5} for Toxcom and OLID respectively. Decreasing the rate too low results it worse performance, as seen in the last experiment.

The results are better than all non domain adaptation approaches. This could be attributed to the conversion scheme stated above, which leads to an increase in relevant training data. Toxcom and OLID are similar datasets, and using the schema stated above, we are able to exploit the similarity between these two datasets. On the other hand, the results are worse than the standard domain adaptation approach. This could be due to the fact that the conversion rules are manually created. Manually created rules are unable to take into consideration every situation and can lead to simplifications. The mapping used above is unable to perfectly convert the labels between datasets.

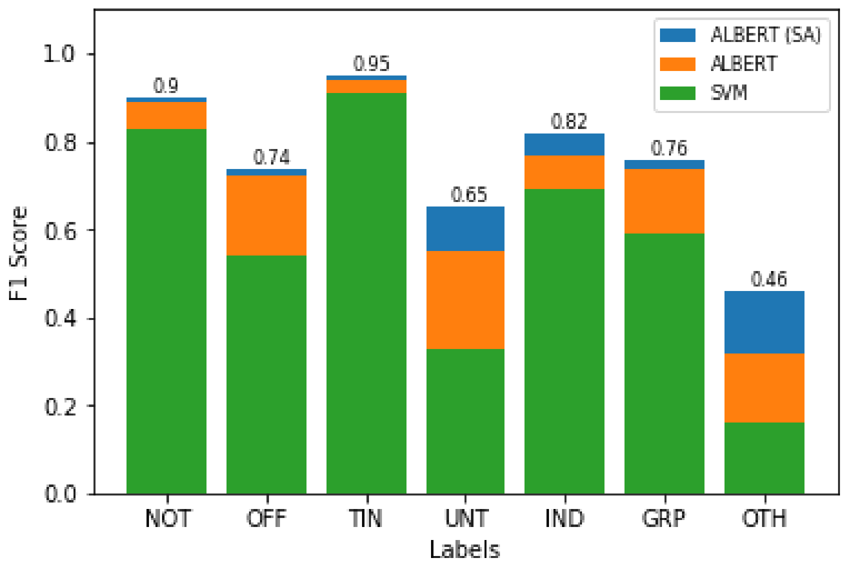

Figure 4: Classwise F1 scores across all 3 levels. The most significant improvements are on classes which originally had poor performance.

Analysis of Domain Adaptation

ALBERT with standard domain adaptation, ALBERT (SA), is the overall best performing model. We compare its performance with the baseline SVM, and best performing model without domain adaptation, ABLERT.

The most notable improvement is on *OTH* samples. ALBERT (SA) has an F1 score of 0.46, which is an improvement of 0.3 points or 187.5% over SVM and 0.14 points or 43.75% over ALBERT. Two things are unique to the *OTH* class. Firstly, it contains of only 395 training samples, which is extremely low. Next, it is also a broad and vague category, containing of text categorized as miscellaneous. ALBERT (SA) efficiently makes use of the additional training on Toxcom to help it on these samples. Significant performance gains are also observed on *UNT* samples, where ALBERT (SA) improves ALBERT's score of 0.55 to 0.65, which is an improvement of 0.1 points or 18%.

Conversely, performance on classes on which ALBERT already has high F1 scores, such as *NOT* and *TIN*, do not see major improvements through domain adaptation. On *NOT* samples, ALBERT (SA) improves only 0.01 points or 1.11%. On *TIN* samples, ALBERT (SA) improves only 0.01

points or 1.06% over ALBERT. From the above observations, it is clear that domain adaptation most significantly benefits the classes with low number of training samples and low initial performance.

Interestingly, *IND* and *GRP* samples go against this trend. *IND* has more training samples than *GRP*. ALBERT also has better performance on *IND* than *GRP*. In spite of this, ALBERT (SA) shows a bigger improvement on *IND* samples compared to *GRP*. To analyze this, we can refer to Table 4. Toxcom consists of 7,877 *insult* samples, 478 *threat* samples and 1,405 *identify hate* samples. *Insult* and *threat* can be roughly compared to *IND* samples and *identity hate* to *GRP* samples. Thus, domain adaptation has exposed the model to 8,289 new *IND* like samples and only 1,405 new *GRP* like samples. This results in domain adaptation improving the performance on *IND* samples more than *GRP* samples.

CHAPTER 5

CONCLUSION

Transformer based methods such as BERT and ALBERT are found to perform the best on OLID. ALBERT has a much smaller memory requirement than BERT, allowing the use of larger and more complex ALBERT models. This, in turn, leads to ALBERT outperforming BERT on the task at hand.

Two approaches to domain adaptation are taken. The standard domain adaptation approach proves to be more effective than the source label adjustment approach, while also eliminating the need for domain knowledge and manual creation of a conversion schema.

A detailed analysis of the improvements due domain adaptation leads to a few observations. First, domain adaptation significantly benefits underrepresented and underperforming classes. Domain adaptation improves the scores of the worst performing class by 43.75%, while improving only 1.06% on the best performing class. Next, domain adaptation can also significantly benefit classes which are already well represented and have good performance, provided there is enough similar data in the source domain.

The availability of high quality labeled data is the prohibitive factor in many tasks. Domain adaptation opens up a number of new possibilities for these tasks. Our experiments prove that domain adaptation can significantly help performance in cases where data is sparse. For future work, different domain adaptation methodologies can be tested. Additionally, using multiple source datasets, instead of just one, is an interesting direction to explore.

REFERENCES

[1] Alexander T. Vazsonyi et al. "Cyberbullying in context: Direct and indirect effects by low self-control across 25 European countries". In: *European Journal of Developmental Psychology* 9.2 (2012), pp. 210–227. DOI: 10.1080/17405629.2011.644919. eprint: https://doi.org/10.1080/17405629.2011.644919. URL: https://doi.org/10.1080/17405629.2011.644919.

[2] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882 (2014). arXiv: 1408.5882. URL: http://arxiv.org/abs/1408.5882.

[3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2018. arXiv: 1810.04805 [cs.CL].

[4] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.* 2019. arXiv: 1909.11942 [cs.CL].

[5] Jigsaw/Conversation AI. "Toxic Comment Classification Challenge". In: 2018.

[6] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems.* 2013, pp. 3111–3119.

[7] Jeffrey Pennington, Richard Socher, and Christopher 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.

[8] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[10] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

[11] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *CoRR* abs/1906.08237 (2019). arXiv: `1906.08237`. URL: `http://arxiv.org/abs/1906.08237`.

[12] Alex Nikolov and Victor 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`.

[13] Jiahui Han, Shengtan Wu, and Xinyu 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`.

[14] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *CoRR* abs/1703.04009 (2017). arXiv: `1703.04009`. URL: `http://arxiv.org/abs/1703.04009`.

[15] Vikas S Chavan and SS Shylaja. "Machine learning approach for detection of cyber-aggressive comments by peers on social media network". In: *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI).* IEEE. 2015, pp. 2354–2358.

[16] Ping Liu, Wen Li, and Liang 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`.

[17] Dietrich Klakow. *Offensive Language Detection with Neural Networks for Germeval Task 2018.* de. Ed. by Josef Ruppendorfer - Melanie Siegel - Michael Wiegand (Hrsg.) Online available:

https://epub.oeaw.ac.at/?arp=0x003a10e4 - Last access:9.5.2020. Wien, 2018. URL: https://epub.oeaw.ac.at/?arp=0x003a10e4.

[18] Spiros V. Georgakopoulos et al. “Convolutional Neural Networks for Toxic Comment Classification”. In: *CoRR* abs/1802.09957 (2018). arXiv: 1802.09957. URL: http://arxiv.org/abs/1802.09957.

[19] Pranav Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* 2016. arXiv: 1606.05250 [cs.CL].

[20] Marcos Zampieri et al. “Predicting the Type and Target of Offensive Posts in Social Media”. In: *Proceedings of NAACL*. 2019.