Expanding the training data for neural network based hate speech classification

Ashwin Geet d'Sa

UNIVERSITÉ DE LORRAINE

Thèse de Doctorat

Ashwin Geet D'SA

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Lorraine
Mention Informatique

École doctorale : IAEM

Unité de recherche : Laboratoire Lorrain de Recherche en Informatique et ses Applications UMR 7503

Soutenue le 6 Mai 2022
Thèse N°:

Expanding the training data for neural network based hate speech classification

JURY

Directrice de thèse :	**Irina ILLINA**, HDR, Maître de conférence, Université de Lorraine, LORIA-INRIA, France
Co-directeur de thèse :	**Dominique FOHR**, Chargé de Recherche, CNRS, LORIA-INRIA, France
Rapporteur :	**Richard DUFOUR**, Professeur, Université de Nantes, LS2N, France
Rapporteur :	**Pavel KRÁL**, Professeur associé, University of West Bohemia, République Tchèque
Examinateur : (et président du jury)	**Georges LINARÈS**, Professeur, Université d'Avignon, LIA, France
Examinateur :	**François PORTET**, Professeur, Université Grenoble Alpes, LIG, France
Examinatrice :	**Josiane MOTHE**, Professeur, Université de Toulouse, IRIT, France
Examinateur :	**Christophe CERISARA**, HDR, Chargé de Recherche, CNRS, LORIA-INRIA, France
Invité :	**Dietrich KLAKOW**, Professeur, Universität des Saarlandes, LSV, Allemagne
Invité :	**Angeliki MONNIER**, Professeur, Université de Lorraine, CREM, France

Résumé

L'augmentation phénoménale de l'utilisation d'Internet, qui permet la diffusion d'opinions, a également entraîné une augmentation des discours de haine en ligne. Les discours de haine sont des comportements de communication antisociaux, qui conduisent à des menaces ou à des violences envers un individu ou un groupe. Les modèles basés sur l'apprentissage profond sont devenus la solution état de l'art pour détecter les discours de haine. Cependant, la performance de ces modèles dépend de la quantité de données d'entraînement étiquetées. Dans cette thèse, nous explorons différentes solutions pour augmenter les données d'entraînement afin de d'entrainer un modèle performant pour la classification des discours de haine.

Comme première approche, nous proposons d'utiliser *apprentissage semi-supervisé* pour combiner une grande quantité de données non étiquetées, facilement disponibles sur Internet, avec une quantité limitée de données étiquetées pour entraîner un classifieur. Pour cela, nous utilisons l'algorithme de propagation d'étiquettes. La performance de cette méthode dépend de l'espace de représentation des données. Nous montrons que les plongement (embeddings) de phrases pré-entraînés sont agnostiques et donnent de mauvais résultats. Nous proposons une approche simple et efficace basée sur les réseaux de neurones pour transformer ces représentations pré-entraînées en représentations adaptées à la tache de détection de la haine. Cette méthode permet d'améliorer considérablement les performances dans des scénarios à faibles ressources. Nous explorons les trois approches proposées dans des scénarios à faibles ressources et nous montrons qu'elles permettent d'améliorer considérablement les performances dans des configurations à très faibles ressources.

Dans notre deuxième approche proposée, nous explorons une méthode *d'augmentation de données (data augmentation)*, une solution pour générer des échantillons synthétiques en utilisant les données d'entraînement originales. Notre technique d'augmentation des données est basée sur un modèle de langage conditionnel GPT-2 ajusté (fine-funing) sur les données d'entraînement originales. Notre approche utilise un modèle BERT pour sélectionner des données synthétiques de bonne qualité. Nous étudions l'effet de la quantité de données générées et montrons que l'utilisation de quelques milliers d'échantillons synthétiques permet d'améliorer considérablement les performances de la classification des discours haineux. Notre évaluation qualitative montre l'efficacité de l'utilisation de BERT pour filtrer les échantillons générés.

Dans notre approche finale, nous utilisons *l'apprentissage multi-tâches* comme méthode pour combiner plusieurs corpus disponibles de discours haineux et entraîner conjointement un seul modèle de classification. Notre approche exploite les avantages d'un modèle de langage pré-entraîné (BERT) pour les couches partagées de notre architecture multi-tâches. Nous considérons un corpus de discours de haine comme une tâche. Ainsi, nous

adaptons le paradigme de l'apprentissage multi-tâches à l'apprentissage multi-corpus. Nous montrons que le réglage fin du modèle multi-tâches pour un corpus spécifique permet d'améliorer les résultats. De plus, notre méthode a obtenu de bonnes performances dans le cadre de l'adaptation au domaine (domain adaptation).

Abstract

The phenomenal increase in internet usage, catering to the dissemination of knowledge and expression, has also led to an increase in online hate speech. Online hate speech is anti-social communicative behavior, which leads to the threat and violence towards an individual or a group. Deep learning-based models have become the state-of-the-art solution in classifying hate speech. However, the performance of these models depends on the amount of labeled training data. In this thesis, we explore various solutions to expand the training data to train a reliable model for hate speech classification.

As the first approach, we use a *semi-supervised learning* to combine the huge amount of unlabeled data, easily available on the internet, with a limited amount of labeled data to train the classifier. For this, we use the label-propagation algorithm. The performance of this method depends on the representation space of labeled and unlabeled data. We show that pre-trained sentence embeddings are label agnostic and yield poor results. We propose a simple and effective neural-network-based approach for transforming these pre-trained representations to task-aware ones. This method achieves significant performance improvements in low-resource scenarios.

In our second approach, we explore *data augmentation*, a solution to obtain synthetic samples using the original training data. Our data augmentation technique is based on a single conditional GPT-2 language model fine-tuned on the original training data. Our approach uses a fine-tuned BERT model to select high-quality synthetic data. We study the effect of the quantity of augmented data and show that using a few thousand synthetic samples yields significant performance improvements in hate speech classification. Our qualitative evaluation shows the effectiveness of using BERT for filtering the generated samples.

For our final approach, we use *multi-task learning* as a method to combine several available hate speech datasets and jointly train a single classification model. Our approach leverages the advantages of a pre-trained language model (BERT) as shared layers of our multi-task architecture. We treat one hate speech corpus as one task. Thus, adopting the paradigm of multi-task learning to multi-corpus learning. We show that training a multi-task model with several corpora achieves similar performance as training several corpus-specific models. Nevertheless, fine-tuning the multi-task model for a specific corpus allows improving the results. We demonstrate the effectiveness of our multi-task learning approach for domain adaptation on hate speech corpora.

We explore the three proposed approaches in low-resource scenarios and show that they achieve significant performance improvements in very low-resource setups.

Acknowledgements

Although it's well said that a thesis is an individual's work, I would argue that it's the little or giant help from others that leads to its completion. During the journey of this thesis, I was feeble to walk without the support of walking sticks. Hence, I would like to thank everyone who has directly or indirectly contributed their share of being my walking stick for the completion of this work.

First and foremost, I would like to thank my supervisors, Irina Illina and Dominique Fohr. They have generously contributed through their enormous time and efforts in directing me towards achieving my thesis goals. I appreciate that they were not just a source of professional help, but also keen on helping with personal needs. I am thankful for all the support and help offered by them. I would like to thank Dietrich Klakow for enabling my research visit to Saarland University, following his continued supervision till the end of my thesis. I would like to thank Richard Dufour and Pavel Král for reviewing my thesis and providing valuable feedback. I would extend my gratitude to Georges Linarès, François Portet, Josiane Mothe, Christophe Cerisara, and Angeliki Monnier for being part of the thesis jury committee. The questions projected by the members of the jury have broadened my vision of this research field. Special thanks to Dana Ruiter for frequent scientific discussions that have helped me in shaping my research work. "*We stand on the shoulders of giants*", I would like to thank all the researchers (*giants*), whose prior work has contributed to enriching my knowledge in this field.

I would like to thank the members of the MULTISPEECH team of LORIA-INRIA for the entire ecosystem during my 3 years of tenure as a PhD student. It was nice to have technical discussions with various members of the team, and also memories that I always cherish. Few of the members have welcomed me to Nancy and the team, while few of them have provided me with their extended support. I would like to thank by naming a few- Denis Jouvet, Emmanuel Vincent, Mostafa Sadeghi, Vincent Colotte, Tulika Bose, Sandipana Dowerah, Seyed Hossein, Vinicius Ribeiro, Nicolas Zampieri, Marina Krémé, Louis Delebecque, Colleen Beaumard, Ajinkya Kulkarni, Manuel Pariente, Sunit Sivasankaran, Brij Srivastava, Nicolas Turpault, Nicolas Furnon, Élodie Gauthier. One such amazing place in this team was my office. I would like to thank my office mates- Louis Abel, Michel Olvera, Shakeel Ahmad Sheikh, Joris Cosentino, Raphaël Duroselle, Md Sahidullah for making it "the best and most lively office", from having lots of white-board discussions to chess, table-tennis, wor(l)dle, etc.

I would like to thank Hélène Cavallini, Delphine Hubert, Souad Boutaguermouchet, Aurore Tranchina, and Sabrina Ferry-Tritz for the administrative support.

I would like to thank the financial source for my PhD, the M-PHASIS project supported by the French National Research Agency (ANR) and German National Research Agency (DFG) under contract ANR-18-FRAL-0005. The presented in this thesis were carried

out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations.
I would like to thank my teachers BG Prasad and Indiramma M from BMS College of Engineering, and Sampath S and Nanjesh BR from Adichunchanagiri Institute of Technology for encouraging and supporting me to pursue my PhD.
With my deepest gratitude, I would like to thank my family for their unconditional support. Especially, my parents, James D'Sa and Gladys Saldanha for they have always had faith in me in everything that I have done so far, and being my backbone throughout. I would extend my gratitude to my uncles Vinod Lobo and Sunil Saldanha, and my aunt Shalini D'Souza. I would also thank all my friends for they have been through my journey of good times, bad times, and hard times.
Last but never least, my special and deepest thanks to Imran Sheikh. His time in terms of whiteboard and technical discussions, and internal reviews of my articles were one of the main reasons for the progress of my thesis. He has also been my moral and personal support during the journey of my PhD.

Contents

List of figures

List of tables

List of acronyms

BART bidirectional and auto-regressive transformers
BERT bidirectional encoder representations from transformers
Bi-LSTM bidirectional long short-term memory
BOW bag of words
CBOW continuous bag of words
C-GRU convolutional-gated recurrent unit
CNN convolutional neural network
DAN deep averaging network
DNN deep neural network
ELMo embeddings from language models
GloVe global vectors for word representation
GPT generative pre-trained transformer
GRU gated recurrent unit
LSTM long short-term memory
MCL multi-corpus learning
MLP multi-layer perceptron
NER named entity recognition
NLP natural language processing
OOV out-of-vocabulary
PCA principal component analysis
POS part-of-speech
ReLU rectified linear unit
RoBERTa robustly optimized BERT approach
RNN recurrent neural network
SCL single-corpus learning
SMOTE synthetic minority over-sampling technique
SVMs support vector machines
USE universal sentence encoder
VAE variational auto-encoder

1 Introduction

1.1 Motivation

The extraordinary increase in connectivity established through the internet has facilitated several daily life activities such as easy access to information in the form of educational resources or news, faster communication through email or conferencing tools, work collaborations, entertainment, etc. One notable advantage is encouraging people to express their opinions through special interest forums, news forums, social media, etc. While the internet has many advantages, few people misuse the internet by spreading false information, hate speech, etc.

Hate speech is anti-social communicative behavior. It targets the minority section of the society based on gender, religion, ethnicity, etc. Hate speech leads to humiliation, threat, or violence towards an individual or a group (Delgado and Stefancic, 2014). Earlier, hate speech propagated in printed and oral forms. However, with the advent of the internet and the increase in people gathering online, social media platforms and other online forums have become common grounds for the propagation of hate speech, leading to the acceleration in the spread of hate speech. One such example is the rise in hate towards people of Asian origin through social media due to the outburst of the 'COVID-19' pandemic, which led to harassment, discrimination, and physical assaults (Yu et al., 2020).

Online hate speech is prohibited by law in many counties. Social media or other online platforms are held accountable for the hateful content posted on them. Failure to remove hateful content on time leads to lower reputation of the platform, loss of users, stock market loss[1], and penalty[2] from the legal authority. Hence, online platforms need to remove the hateful content posted by their users. However, due to the massive flow of content, manual monitoring and moderating the online content is time-consuming and very expensive. This demands automatic hate speech detection systems, which can be built using machine learning and Natural Language Processing (NLP) techniques.

[1] https://www.business-standard.com/article/international/mark-zuckerberg-loses-7-billion-as-companies-boycott-facebook-ads-120062701217_1.html

[2] https://www.cnet.com/tech/services-and-software/german-hate-speech-law-goes-into-effect-on-1-jan/

1.2 Framework of M-PHASIS

The work presented in this thesis is conducted in the framework of "**M**igration and **P**atterns of **Ha**te **S**peech **i**n **S**ocial Media - A Cross-Cultural Perspective" (M-PHASIS) project. This project is jointly funded by 'Agence Nationale de la Recherche' (National Agency for Research, ANR) and 'Deutsche Forschungsgemeinschaft' (German Research Foundation, DFG). M-PHASIS is an inter-disciplinary project with participants from social science and computer science backgrounds. The work is conducted as a collaboration between four institutes, namely, Centre de recherche sur les médiations (CREM), Université de Lorraine, France; Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA), Université de Lorraine, France; Johannes Gutenberg-Universität (JGUM), Mainz, Germany; and Saarland University (SAAR), Saarbrücken, Germany. While CREM and JGUM focus on the social science aspects of this project, LORIA and SAAR focus on computer science aspects.
From the social science perspective, the project aims to advance the understanding of hate speech towards migrants in France and Germany and perform a cross-cultural comparison in the prevalence of hate speech among these two countries. From the computer science perspective, the project aims to build a hate speech detection system that can automatically detect hateful content in text. The project aims to collaboratively collect and annotate real-life examples through news forums and social media outlets. As a participating institute, we at LORIA have emphasized on improving hate speech classification models.

1.3 Expanding training data for hate speech classification

The classical hate speech classification systems used lexical features as input to the models. A simple hate speech classifier was built using a dictionary of abusive words and hate words Razavi et al. (2010). It was extended to identify the semantic relationship among the words Xu and Zhu (2010); Gitari et al. (2015) using dependency parsers and part-of-speech (POS) tagging. As the research in NLP advanced, various features such as word n-grams and character n-grams (Nobata et al., 2016), bag of words (BOW), etc, were typically used with non-deep-learning-based classifiers such as logistic regression, Support Vector Machines (SVMs)(Cortes and Vapnik, 1995), etc (Chen et al., 2012; Waseem and Hovy, 2016; Nobata et al., 2016).
With the evolution of deep learning, and its application in NLP, Badjatiya et al. (2017) showed that deep-learning-based classifiers such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) outperformed traditional machine learning-based classifiers such as SVMs and logistic regression. Following this, many researchers have shown that deep learning-based techniques work better than traditional machine learning techniques for hate speech classification (Del Vigna et al., 2017; Lee et al., 2018; Park and Fung, 2017; Gambäck and Sikdar, 2017).
Deep learning-based pre-trained language models proved to be the state-of-the-art solutions for a range of NLP tasks. These models are trained on huge unlabeled corpora and

can be fine-tuned on downstream tasks. Recently, many researchers have adopted fine-tuning the models like Embeddings from Language Models (ELMo)(Peters et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT)(Devlin et al., 2019) for hate speech detection tasks (Banerjee et al., 2020; Bodapati et al., 2019).
Due to the recent advancements in deep learning techniques, throughout this thesis, we have focused on *hate speech classification using deep neural network (DNN) based classifiers*.
The performance of DNN-based classifiers depends on the amount of labeled data available to train them. However, due to the time and cost involved in annotation, many hate speech datasets have very few amounts of labeled samples. To increase the number of labeled samples to train a reliable classifier, we explore three different ways by answering the following research questions:

1. How to efficiently combine the huge amount of unlabeled data along with the labeled training data to train a better classifier?
2. How to generate new data from the original training data?
3. How to combine several available hate speech corpora to increase the total number of training samples?

Further in this section, we describe the different approaches used to address the above research questions.

1.3.1 Semi-supervised learning

Semi-supervised learning allows to combine a small amount of labeled data with a large amount of unlabeled data during training to improve the classification model's performance (Abney, 2007). In this thesis, we use *semi-supervised learning* to address our first research question *"How to efficiently combine the huge amount of unlabeled data along with the labeled training data to train a better classifier?"*. In Chapter 4, We explore semi-supervised learning based on the label propagation (Xiaojin and Zoubin, 2002) algorithm for the hate speech classification task. Label propagation is a graph-based semi-supervised learning technique similar to the k-nearest-neighbours algorithm. This algorithm assumes that data points are close to each other if they are similar, hence, tend to have a similar label. The algorithm relies on the representation of these data points to create a distance graph that captures the proximity of the samples. Thus, choosing the appropriate representation of data points becomes a key point of this algorithm. Pre-trained sentence representations are one such generic representations, where the sentences that are semantically related are close to each other. Thus, we evaluate pre-trained sentence embeddings as representations for label propagation. However, as pre-trained embeddings are generic, and do not contain the task-specific information, we transform this generic representation to task-specific representations. Our approach uses a multi-layer perceptron trained on a very small amount of labeled data to transform these pre-trained representations to task-specific representations. Furthermore, we simulate very low-resource scenarios by reducing the amount of labeled data to show the effectiveness of our semi-supervised learning approach.

1.3.2 Data augmentation

One group of approaches to increase the number of training samples is by exploiting the available training data itself. It involves performing minor perturbation or transforms to the original training data to create new synthetic data. Such perturbations or transformations adds new information to the training data that can benefit the classifier. A range of data augmentation techniques is explored in various NLP applications to improve the model performance. Given the significant advances in the usage of pre-trained models for various NLP applications, in Chapter 5, we explore pre-trained language model-based synthetic sample generations for data augmentation . We use *data augmentation* as a solution to address our second research question "How to generate new data from the original training data?". The proposed approach involves using a small number of labeled samples to train a GPT-2 model using the objective of conditional language modeling and generating data that is close to our training data. We use these synthetically generated samples along with the original training data to train a model to improve the hate speech classification performance. We evaluate the influence of the quantity of the generated samples used for data augmentation and the quality of the generated samples. Furthermore, we reduce the amount of training data to study the behavior of language model-based data augmentation in low-resource scenarios.

1.3.3 Multi-corpus learning

The increase in research interest towards hate speech classification in recent years has led to an increase in the number of available hate speech corpora. Although there are several different corpora available in the domain of hate speech, they differ in terms of the source, annotation schemes, the sampling strategy, the time frame during which the samples are collected, the definition of class labels, etc. In Chapter 6, we use the multi-corpus learning approach based on the paradigm of multi-task learning to combine these corpora to improve the model performance. Multi-task learning aims to improve a model by jointly learning multiple related tasks. Multi-task learning combines the domain-specific information of related tasks to benefit the generalization of the model for all the tasks (Caruana, 1997). In our approach, we treat each corpus as a task. By combining the datasets of related tasks to jointly train a model, multi-task learning creates an implicit data augmentation, reduces the data sparsity issue, and data-dependent noise (Ruder, 2017; Worsham and Kalita, 2020). We use *multi-corpus learning* as a solution to address our third research question "How to combine several available hate speech corpora to increase the total number of training samples?". Our multi-corpus learning approach uses the pre-trained BERT model to benefit from large knowledge learned by BERT during the pre-training. Furthermore, we also evaluate the efficacy of multi-corpus learning in low-resource scenarios. We also use multi-corpus learning as an approach for domain adaptation.

1.4 Contributions of this thesis

In this thesis, we explore different approaches to expand the training data to improve the performance of DNN-based hate speech classification. We also study the effectiveness of these approaches in low resource scenarios. The contributions of this thesis are:

1. We perform *semi-supervised* learning using label propagation to combine unlabeled data with the labeled data. We also propose a technique to transform the pre-trained representations to task-specific representations to increase the effectiveness of the label propagation algorithm. We evaluate and compare our approach on the unlabeled data obtained from the same distribution as well as different distribution as that of the labeled data.
2. We perform data augmentation by generating synthetic samples. We use the training data to train a conditional language model and generate synthetic data. We add this generated data to the original training set to train a classifier. We also perform an extensive study on the influence of quantity and quality of generated data.
3. We aim to improve the classification performance by using the several available hate speech corpora. We combine several hate speech datasets to train a model using the *multi-corpus learning* approach. We perform an extensive evaluation of our approach in very low-resource scenarios. We also extend this work for supervised domain adaptation.

Parts of work described in this thesis have been published in the following articles:

1. **Ashwin Geet D'Sa**, Irina Illina, Dominique Fohr. BERT and fastText embeddings for automatic detection of toxic speech. In *International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*, 2020
2. **Ashwin Geet D'Sa**, Irina Illina, Dominique Fohr. Towards non-toxic landscapes: Automatic toxic comment detection using DNN. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, LREC*, 2020
3. **Ashwin Geet D'Sa**, Irina Illina, Dominique Fohr, Dietrich Klakow, Dana Ruiter. Label propagation-based semi-supervised learning for hate speech classification. In *Proceedings of the First Workshop on Insights from Negative Results in NLP, EMNLP*, 2020
4. **Ashwin Geet D'Sa**, Irina Illina, Dominique Fohr. Classification of hate speech using deep neural networks. Revue d'Information Scientifique & Technique, 25(01), 2020
5. **Ashwin Geet D'Sa**, Irina Illina, Dominique Fohr, Dietrich Klakow, Dana Ruiter. Exploring conditional language model based data augmentation approaches for hate speech classification. In *Text, Speech, and Dialogue: 24th International Conference Proceedings*, 2021

1.5 Organization of the thesis

This thesis is organized as follows:

Chapter 2 describes the various challenges encountered in the hate speech classification and the state-of-the-art solutions used to address these challenges.

Chapter 3 describes the datasets, text preprocessing, and classification metrics used in this thesis. We also present various DNN based embeddings and models used in this thesis.

Chapter 4 deals with our proposed methodology of semi-supervised learning using the label propagation algorithm.

Chapter 5 deals with our proposed methodology of data augmentation approach using the conditional language modeling objective.

Chapter 6 deals with our proposed methodology of multi-task learning using several hate speech corpora.

Chapter 7 presents the results and limitations of the works presented in this thesis. We also provide the perspectives and directions for future research in this chapter.

2 State of the art

2.1 Introduction

Research interest towards hate speech is increasing day by day. However, due to the complex nature of hate speech, building a reliable hate speech classification system remains challenging due to the wide range of problems. Although the objective of this thesis is not to solve all the challenges in hate speech classification, it is important to study the complete panorama of the hate speech domain and the different challenges that exist in this domain from the computer science perspective. This also provides us with directions for various possible research in the field of hate speech classification that can be addressed in the future. In this chapter, we present the challenges encountered in hate speech and the various state-of-the-art solutions used to address these challenges. First, we explain the main problem of an unclear definition of hate speech in Section 2.2. We explain the factors creating the biases and solutions that are used to mitigate them in Section 2.3. The problems associated with diverse labeling schemes are discussed in Section 2.4. We present the problem of implicit and explicit hate speech in Section 2.5. In Section 2.6, we describe how the context influences hate speech, and the solutions used to tackle this problem. Following this, in Section 2.7 we describe the problem of evolving hate speech. In Section 2.8, we describe the problems with misspelled words and other adversaries that can be induced to spoof a hate speech classifier. Finally, we describe the problems of class imbalance in Section 2.9.

2.2 Definition of hate speech

One of the key problems in hate speech classification or annotating hate speech datasets is defining the term "hate speech". A well-defined term helps the annotators to improve the quality of data labeling (GuillermoCarbonell and MichaelWojatzki, 2016) and the dataset itself. There is no precise definition of the term "hate speech" (Davidson et al., 2017; Schmidt and Wiegand, 2017; GuillermoCarbonell and MichaelWojatzki, 2016; Fortuna et al., 2020). The governing bodies that monitor hate speech provide their own definition of hate speech. Furthermore, these definitions are often used as a basis to redefine the term hate speech by the researchers or the authors of datasets. In Table 2.1, we provide the definitions used by governing bodies such as European Union (EU) Council[1]

[1] https://ec.europa.eu/info/sites/default/files/code_of_conduct_hate_speech_en.pdf

and United Nations[2], social media companies such as Facebook[3] or Twitter[4], and the definitions used by researchers while annotating the datasets.

Origin	Definition
EU Council	"Publicly inciting to violence or hatred directed against a group of persons or a member of such a group defined by reference to race, colour, religion, descent or national or ethnic origin"
United Nations	"Any kind of communication in speech, writing or behaviour, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, colour, descent, gender or other identity factor"
Facebook	"Direct attack against people — rather than concepts or institutions— on the basis of protected characteristics: race, ethnicity, national origin, disability, religious affiliation, caste, sexual orientation, sex, gender identity and serious disease. Attacks is defined as violent or dehumanizing speech, harmful stereotypes, statements of inferiority, expressions of contempt, disgust or dismissal, cursing and calls for exclusion or segregation."
Twitter	"You may not promote violence against, threaten, or harass other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease."
Nockleby (2000)	"Any communication that disparages a target group of people based on some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic"
Nobata et al. (2016)	"Language which attacks or demeans a group based on race, ethnic origin, religion, disability, gender, age, disability, or sexual orientation/gender identity"
Davidson et al. (2017)	"language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group."

Table 2.1: Definition of hate speech from various origin.

From Table 2.1, we observe that the definition provided or used by the governing bodies, the social media organizations, or the authors are not the same. However, they have similarities, where most of the definitions emphasizes hate speech as a target towards an individual or a group. The definitions provided by EU Council, Facebook, Twitter specifically treat hate speech as content that leads to violence or attacks, however,

[2] https://www.un.org/en/genocideprevention/documents/advising-and-mobilizing/Action_plan_on_hate_speech_EN.pdf
[3] https://www.facebook.com/communitystandards/hate_speech
[4] https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

the definition provided by United Nations, Nockleby (2000), Nobata et al. (2016), and Davidson et al. (2017) treat it as content that leads to humiliation.
Hate speech detection can be treated as a classification task. The automatic text classification techniques using Natural Language Processing (NLP) or machine learning relies on the quality of data. Improved annotation guidelines lead to higher quality data, which in turn improves the features and the representations captured by these classifiers to better characterize hate speech data. Thus, it is important to have a well-formulated definition of hate speech to provide clear guidelines for the annotators to improve the data reliability.
GuillermoCarbonell and MichaelWojatzki (2016), claimed that initial annotations had a very low inter-annotator agreement score. A discussion among the annotators revealed that the unclear definition of hate speech led to poor annotation quality. Fortuna et al. (2020), showed that the class label definition of various terms related to hate speech, such as 'hate speech', 'abuse', 'sexism', etc. varied across datasets, thus datasets with the same labels were not homogeneous. Therefore, the trained models lost the generalization potential and a model trained on one corpus gave a poorer performance on other corpora. Fortuna et al. (2020) also provided the guidelines for future annotation, which recommended using a clear and distinctive definition of class labels. Thus, to eliminate the problem of an unclear definition of hate speech Davidson et al. (2017), Founta et al. (2018), and Basile et al. (2019) formulated and provided a clearer definition of the term 'hate speech' to the annotators. Similarly, in addition to the definition of 'hate speech', Waseem and Hovy (2016) provided a set of guidelines in the form of rules and example tweets to the annotators for improving the annotation quality.

2.3 Biases in hate speech datasets

The sampling strategy, time frame during which the dataset is collected, demography of the annotators, etc. creates biases in the datasets (Yin and Zubiaga, 2021; Wiegand et al., 2019; Razo and Kübler, 2020). Such unintended biases in the datasets can decrease the performance of the hate speech classifiers. Furthermore, these biases reduce the generalization capabilities of a model trained on one corpus across other corpora. In this section, we discuss biases that are commonly observed in hate speech datasets and the solutions that have been used to mitigate them.

2.3.1 Sampling bias

As the occurrence of hate speech on social media is very low (between 0.1% and 3% on Twitter) (Founta et al., 2018), many authors use a set of hate or abusive keywords to sample data from social media. For example, Davidson et al. (2017) collected samples based on the words from the hatebase[5] lexicon. Such sampling induces significant bias in the corpus as it gives an unrealistic class label distribution, where 77.4%, 5.6%, and 16.9% of the samples belonged to 'abusive', 'hate speech', and 'normal' classes, respectively,

[5] https://www.hatebase.org

where samples from 'abusive' class is majority. To prevent such unrealistic class label distribution and to have more abusive and hateful samples than the naturally occurring low numbers, Founta et al. (2018) used boosted random sampling. Here, the samples are pre-selected using a trained sentiment analysis model, because 'hate speech' and 'abusive' samples often contain negative sentiments. This gives 27.2%, 4.5%, 53.8%, and 15.5% of 'abusive', 'hate', 'normal', and 'spam' samples, respectively. Thus, the boosted random sampling maintains a lesser biased dataset by having more samples for normal speech while having sufficient number of samples for the minority hate speech.
Waseem and Hovy (2016) sampled tweets using racial and sexual slurs along with hand-picked hashtags and terms that provoked sexist tweets. Wiegand et al. (2019) showed that most of the abusive tweets collected by Waseem and Hovy (2016) had a strong correlation with sports-related words such as 'commentator', 'football', 'announcer', etc. Additionally, they showed this kind of non-random sampling in this dataset led to samples being collected based on topics such as "role of women in sports". This led to false correlation of words associated with football with hate class. The authors showed that the classifier trained on the Waseem and Hovy (2016) dataset misclassified 70% of the non-abusive tweets containing the words 'football' and 'sport' as abusive. This kind of bias was termed **topic bias**. Furthermore, Wiegand et al. (2019) compared the boosted random sampling strategy adopted by Founta et al. (2018) and showed that unlike Waseem and Hovy (2016) the abusive tweets collected by Founta et al. (2018) had a strong correlation with abusive words such as 'bitch', 'niggas', 'fucking', etc. Wiegand et al. (2019) proposed cross-domain testing (a technique where the model is trained on one dataset and tested on another dataset) as a measure to evaluate the bias in the datasets. They showed that datasets of Waseem and Hovy (2016),Warner and Hirschberg (2012), and Kumar et al. (2018a) collected using biased sampling showed lower cross-domain performance compared to the datasets of Founta et al. (2018),Wulczyn et al. (2017), and Razavi et al. (2010) collected using boosted random sampling.

2.3.2 Author bias

The authors of the tweets or hateful posts can influence the performance of the classifiers. This is because the style of writing varies across users and if significant samples from a particular class come from the same author, then the classifier may capture the author's writing style rather than capturing the characteristics of the contents. For example, in the dataset provided by Waseem and Hovy (2016), more than 70% of the sexist tweets came from just two authors and 96% of the racist tweets came from a single author (Qian et al., 2018; Arango et al., 2020). However, assuming that abusive tweets are usually posted by a few set of authors, Qian et al. (2018) used such user information to obtain tweets history for authors, filtered them based on text similarity. The filtered tweets were used then along with the original training set to improve the classification performance. Mishra et al. (2018a) used meta-data of users to create a graph, where authors were represented as nodes, and one author following another gives an edge between the nodes. This graph was used to obtain node2vec (Grover and Leskovec, 2016) embedding, which was used as an additional input feature for the classifier. This resulted in improved performance

of classification. A similar study of using author information is shown by Chopra et al. (2020). However, the models trained on such datasets show poor generalization across other datasets (Wiegand et al., 2019).

2.3.3 Annotator bias

The gender, the ethnicity, the level of expertise of an annotator plays an important role in the annotation procedure. Waseem (2016) compared the annotations performed by amateur annotators (crowd-sourcing) against annotations performed by the experts and showed that training the model using the data annotated by experts significantly outperformed the data annotated by amateur annotators. Al Kuwatly et al. (2020) studied the demographic influence of annotators based on age, gender, native language. The authors showed higher inter-annotator disagreement among female annotators than male annotators. Furthermore, they also showed that the classifier trained with English data annotated by native English speakers outperformed annotations made by the non-native speakers. They also showed that models trained with data annotated by younger annotators (below 30 years) performed better than training on data annotated by older annotators. Wich et al. (2020) created an annotator graph using inter-annotator agreement as similarity measure. Furthermore, they used graph-based community detection (Blondel et al., 2008) to create annotator groups based on annotation behavior. The authors showed that models trained with data from one particular annotator group had different classification performances on test sets belonging to the data annotated by different groups. This showed a possible annotator bias in the dataset. Similarly, Sap et al. (2021) studied the relationship of annotators' social stereotypes with annotation behavior, wherein, they observed that increased competence of an annotator towards a particular social group increased the chance that particular annotator labels a sample mentioning that social group as hate speech. From these works, it is clear that the annotator plays an important role in the annotation procedure and in the quality of data annotation. Thus, training the annotators is important to reduce annotator bias in the hate speech datasets.

2.3.4 Other unintended biases

Based on the sampling strategy and other factors, there can be other unintended biases (such as racial or sexual biases) that may influence classification models. Davidson et al. (2019) evaluated racial bias in the models trained using various hate speech datasets. They used tweets labeled with dialect variations indicating non-Hispanic whites, non-Hispanic blacks, Hispanics, and Asians (Blodgett et al., 2016) to measure the racial bias in the models. They reported that tweets written in black English dialects (African-American English) were more likely to be classified as abusive or hateful than compared to tweets written with white English dialects (general American English). Sap et al. (2019) performed a similar study and gave a similar conclusion. Dixon et al. (2018) observed that the identity terms such as 'homosexual', 'atheist', 'gay', etc. occurred more frequently in the toxic class than in the non-toxic class. They manually identified

the identity terms that occur more frequently in toxic speech than in non-toxic speech. The authors then evaluated the bias by creating a synthetic test set containing these identity terms with an equal amount of samples for the non-toxic class. They showed that the classifiers mispredicted the majority of the non-toxic samples containing such identity terms as toxic. This work was extended to identify gender bias by Park et al. (2018). They compared three techniques to train a debiased classifier: a. debiased word embeddings (Bolukbasi et al., 2016) as an alternate to standard pre-trained word2vec (Mikolov et al., 2013a,b), b. augmenting the training set by identifying the gender-based identity terms in the training set and replacing them with the opposite gender to create new samples (gender-swapping), and c. pre-training the classification model with a less biased hate speech corpora. They observed significant bias mitigation by using a combination of debiased embeddings and data augmentation using gender-swapping methods.

2.4 Diverse labeling schemes

The annotation of the hate speech datasets follows a varied range of labeling schemes that creates differences between datasets and also bring out new challenges in hate speech classification. Some authors use binary annotation scheme with 'toxic speech' and 'non-toxic speech' labels, where 'toxic speech' forms an umbrella term and covers various categories of hate and abuse. Wulczyn et al. (2017) dataset is an example of such datasets. However, an extended version of this dataset, which is part of Kaggle's toxic comment classification challenge [6] uses fine-grained multi-labeling: 'toxic', 'severe toxic', 'obscene', 'threat', 'insult', and 'identity hate', respectively. Hate speech is a punishable offense by law in many countries. Hence, it has to be removed from online platforms. However, using profane language is not a punishable offense. Given such legal implications of hate speech, Davidson et al. (2017) in their corpus separated 'hate speech' from 'offensive language' (but not 'hate speech'), by creating three labels, 'hate speech', 'offensive language', and 'neither' (neither offensive nor hate speech), respectively. Such multi-class classification is more challenging than performing binary classification. Consequently, Founta et al. (2018) followed a similar idea and labeled the data into four classes, namely, 'normal', 'abusive', 'hateful', and 'spam'. However, Waseem and Hovy (2016) in their dataset mainly focused on racism and sexism. They published the dataset with three class labels, namely, 'racism', 'sexism', and 'none'. Similarly, Basile et al. (2019) used a multi-label classification scheme, with 'hate' or 'non-hate', followed by the labels 'group' or 'individual' as targets of hate, and finally, 'aggressive' versus 'non-aggressive', thus creating a multi-labeled corpus. The abusive or offensive tweets in Waseem and Hovy (2016); Basile et al. (2019) are labeled as 'non-hate', thus making the labeling scheme different from Davidson et al. (2017), Founta et al. (2018), and (Wulczyn et al., 2017) datasets, respectively. Similar to Waseem and Hovy (2016), some datasets are thematic, which focus on a particular type of hate such as sexism or misogyny (Fersini et al., 2018; Jha and Mamidi, 2017) or Islamophobia (Chung et al., 2019), etc.

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

The classifiers that are trained on datasets involving broader category such as 'toxic-speech' versus 'non-toxic speech' or 'abusive speech' versus 'normal speech' achieves higher performance than the classifiers that are trained on thematic datasets or more sophisticated datasets for multi-class or multi-label hate speech classification tasks (Fortuna et al., 2021). The differences in the labeling schemes also reduce the generalization capabilities of models trained on one dataset over other datasets (Pamungkas and Patti, 2019; Fortuna et al., 2021).

2.5 Implicit versus explicit hate speech

Hate speech classification is challenging as hate can be expressed as implicitly or explicitly (Waseem et al., 2017b). Explicit hate speech contains words or phrases such as racial or sexual slurs or profane words that can unambiguously express hatred towards a group or an individual. Whereas, implicit hate speech uses sarcasm, irony, or other ambiguous means to express hate. Implicit hate speech is difficult for humans to annotate, as it requires external knowledge or context to understand that the sentence is hateful (van Aken et al., 2018; Davidson et al., 2017; Alatawi et al., 2021; Mozafari et al., 2019). Table 2.2 shows examples of implicit and explicit hate speech.

Table 2.2: Definition of hate speech from various origin.

Explicit hate speech	**Implicit Hate Speech**
This girl is retarded. (Davidson et al., 2017)	*Imagine a zombie scenario where the zombies are mainly googles & the scientists who unleash the googles are skypes.* (Magu et al., 2017)
this guy is the biggest faggot omfg (Davidson et al., 2017)	*"Those guys are the definition of white trash"* (Davidson et al., 2017)
@USER Stfu you annoying ass bitch before i make you hoe (Founta et al., 2018)	*"I will remove all your organs in alphabetical order" I have never seen someone so pleased to say that ever :)* (Founta et al., 2018)

Magu et al. (2017) showed that social media users utilized alternate terms to refer to particular targets of hate. For example, 'Google' and 'Skype' were used as alternate words for 'Black' and 'Jew' respectively. The word 'Skype' that refers to a video conferencing tool is generally used in a non-hate context, for example, *"We enjoyed our conversation on Skype"*. However, when used in a sentence such as *"Gas the Skypes"* (Magu et al., 2017), it refers to hate intended towards the Jewish community. Furthermore, to annotate such comments appropriately, the annotator should be aware of the context of the comment in which the word 'Skype' is used and the historical knowledge of the Holocaust. Davidson et al. (2017) observed that tweets that did not contain any explicit word or phrase indicating hate or abuse were misclassified as 'normal speech' by the classification models. Similarly, hate speech tweets that did not contain any racial

or sexual slur, but contained just profane words were misclassified as 'offensive speech'. van Aken et al. (2018) performed an in-depth error analysis of hate speech classification and observed that hate speech samples expressed using implicit hate(sarcasm, irony, metaphors, absence of slurs, etc.) were common reasons for a model to misclassify hate speech samples. Wiegand et al. (2019) showed that datasets with biased sampling contained a higher amount of implicit abuse, and the models trained on such datasets yielded poor classification performance in cross-domain testing.
As implicit hate speech is hard for the annotators to identify, few works tackled the problem at the annotation level. Kumar et al. (2018b) hired linguistics experts to annotate their data into overt and covert aggression, where overt aggression uses explicit words to express aggression, and it is expressed through rhetoric questions or sarcasm in covert aggression. Similarly, Caselli et al. (2020) re-annotated the 'offensive' class samples from the dataset of OffensEval shared task (Zampieri et al., 2019a,b) into implicit and explicit abuse. The samples were re-annotated by the authors themselves, thus avoiding crowdsourcing. The authors showed that using dictionary-based approaches for classifying abuse versus non-abuse gave poorer classification performance than using a model such as Bidirectional Encoder Representations from Transformers (BERT). Furthermore, the authors showed that modeling 'implicit abuse' is a challenging task even for the BERT model. They performed multi-class classification into 'implicit abuse', 'explicit abuse', and 'non-abuse' classes, respectively, and obtained poor classification performance for the 'implicit abuse' class. ElSherief et al. (2021) showed that the BERT classifier performed better than Support Vector Machines (SVMs)(Cortes and Vapnik, 1995) in presence of implicit hate speech samples, however, the overall performance of implicit hate speech classification was poor. Wang et al. (2020) used representation obtained from a sarcasm detection model (Ghosh and Veale, 2016) as additional input to the hate speech classifier to detect the presence of sarcasm, hence, improving implicit hate speech classification. However, such a model fails to detect non-sarcasm-based implicit hate speech.
The classification of hate speech due to implicit hate speech remains an open and challenging problem. In this thesis, we do not address this problem.

2.6 Influence of context

Sometimes, a sentence can be hateful based on the context information (Schmidt and Wiegand, 2017). The contents of a given sentence itself may act as a context, i.e, a given word can turn the sentence into hateful or non-hateful based on the context in which it is used. For example, the word 'black' can refer to color or ethnicity. In a given sentence, "He wore a black T-Shirt", 'black' refers to the color. However, in the sentence "if you aren't black don't fucking say the n word!!!" (Founta et al., 2018), the word 'black' refers to ethnicity and the sentence intends hate.
Caselli et al. (2020) showed that BERT-based classification outperforms the dictionary-based approach. This is because the BERT model processes the entire sentence and the attention mechanism allows to take into account context information for all the words in the sentence.

Sometimes, the sentence is insufficient to perform the powerful classification and may need additional context such as a previous comment or the article associated with it. Caselli et al. (2020) showed that some comments when read independently were non-hateful, however, when their parent comment was taken into account, comment became hateful. Thus, Caselli et al. (2020); Gao and Huang (2017) provided annotators with the sequence of parent and response comments to consider additional context information, and the dataset was made available with this context information. Pavlopoulos et al. (2020) showed that the absence of context information can reduce the annotation quality as well as the classifier's performance.
A few researchers go beyond textual features and include multi-model features such as images. Gomez et al. (2020) showed that some text comments convey hateful messages only when associated with the images with which it is posted, and thus created a multi-modal hate speech dataset. Furthermore, the text contained in the images can also express hate. The authors extracted the image features using the pre-trained Inception V3 model (Szegedy et al., 2016) and extracted the text from images using the *Google Vision API Text Detection* module.[7] They showed that combining these additional features improved classification performance. A similar study is performed by Kiela et al. (2020). Rezvani et al. (2020) showed that features extracted from images and additional comments considered as context improved the classification performance for cyber-bullying detection.
Context-based and multi-modal hate speech classification is an emerging area in hate speech classification. With very few context-based datasets and research carried out in this domain, it remains an open challenge. In this thesis, we have not considered the datasets providing the image or parent comment context. Thus, we have not incorporated the image or parent comment for hate speech classification.

2.7 Hate speech evolving over time

The hate-related terms and topics evolve and pose new challenges in classifying hate speech (Florio et al., 2020). For example, the word "gay" originally meant "cheerful", but nowadays it is commonly used to refer to "homosexual (male)". Similarly, the flow of hate and the topics associated with hate change with time. Gao et al. (2017) showed a significant increase in the amount of hate speech flowing on Twitter during the US Presidential elections, and many of these hateful Tweets were associated with the election. This poses challenges in creating hate speech datasets, as it creates a topic bias in the datasets. Furthermore, such temporal shifts also reduce the performance of hate speech detection systems. Nobata et al. (2016); Florio et al. (2020) showed that the model trained with data collected at a particular time frame and evaluated over data collected at different time frame gives the reduced performance over new data. Furthermore, they showed that updating the model with the new data improves the model performance over time.

[7] https://cloud.google.com/vision/docs/ocr

The evolving hate speech also reduces the generalization of the model trained on one dataset over the newer datasets (Vidgen et al., 2019; Wich et al., 2021). The changing hate speech vocabulary and the bias induced due to the events occurring at a particular time pose a big challenge to building a reliable hate speech classifier, and this challenge remains little explored.

2.8 Out-of-vocabulary and other adversaries

The words that are present in the test set but absent in the training set create an out-of-vocabulary (OOV) problem. Social media users often use a colloquial writing style or perform intentional misspellings, which forms one of the main reasons for OOV problems. For example, the word 'Fuck' is intentionally written as 'Fcuk' or 'F*ck' to create obfuscation. This reduces classification model performance (Hosseini et al., 2017; Gröndahl et al., 2018). Furthermore, Gröndahl et al. (2018) showed that appending positive words such as 'love' to the hateful sentences also leads to misclassification of the sentence by the classifier. Other forms of inducing adversaries are adding or removing white space, or removing a word or a letter from the word in the sentence (Gröndahl et al., 2018; Oak, 2019).
Few authors have addressed the problem of misspelled words using text pre-processing techniques, where spell-check tools are used to automatically correct the spellings (Pratiwi et al., 2019; Moh et al., 2020; Mishra et al., 2018b). However, simple spell-check tools fail to correct too obfuscated text. Going beyond relying on simple spell-check tools, Hu et al. (2020) represented the text using phonemes, because phonemes of original and misspelled words remained the same. They observed improvements in classification results using phoneme representations. However, such phoneme-based methods will fail when words are written using *leetspeak* ('asshole' written as 'a$$h0l3') (Perea et al., 2008). Character replacement technique such as replacing '$' with 's', '3' with 'e', etc. during pre-processing step is proposed by Gröndahl et al. (2018) for such leetspeak variations.
Few other authors go beyond simple text pre-processing to handle such misspelled or OOV words. Some authors have shown that using character n-grams or character-based models instead of word-based models is more robust to spelling variations (Gambäck and Sikdar, 2017; Wulczyn et al., 2017; Nobata et al., 2016; Mishra et al., 2018b). Furthermore, Mou et al. (2020); Bodapati et al. (2019) showed that using sub-word or word-piece tokenization outperforms character-based and word-based tokenization.

2.9 Class imbalance

The hate speech datasets often have an imbalance in the class label distribution. This is because the occurrence of hate or abusive speech in social media is relatively low (between 0.1% and 3%). Thus, random sampling yields a *natural class imbalanced* dataset with very few hate or abusive speech samples. However, the sampling strategy can create an *artificial class imbalance*. For example, Davidson et al. (2017) collected the tweets based on the hatebase lexicon, which gave a very high percentage of abusive speech

samples (77.4%) than normal speech samples (16.9%). Alternative to key words based sampling, Founta et al. (2018) used boosted random sampling, which gave 53.8% and 27.2% of normal speech and abusive speech samples, respectively. Yet, the percentage of hate speech samples in both these datasets remain relatively low, i.e, 5.6% and 4.9% in Founta et al. (2018) and Davidson et al. (2017) datasets, respectively. Many other hate speech datasets also follow such imbalanced class distributions (Waseem and Hovy, 2016; Wulczyn et al., 2017). In machine learning, the classifiers get overwhelmed by majority class samples and bias the system to favor the prediction of test samples as majority class, thus reducing the performance of the classifier (Chawla et al., 2004).

The class imbalance problem can be mitigated by *downsampling* the majority class samples or by *upsampling* the minority class samples. Downsampling involves removing samples from the majority classes. Downsampling is rarely adopted for hate speech classification. To overcome removing of samples, Rizos et al. (2019) used a modified version of downsampling called *downsampling during training*. In this method, for a given epoch, all the samples from the minority class are selected and for the other classes, the number of samples equal to the minority class is randomly selected. Melton et al. (2020) observed that downsampling during training led to over-fitting of the model to the minority class samples. Hence, along with the downsampling during training, they used ensemble approach. The ensemble approach involved training five classifiers with different random weight initialization and final classification results were taken based on voting (Zimmerman et al., 2018).

Alternative to downsampling, is upsampling. A simple technique to upsample is to replicate the samples from the minority classes to have an equal number of samples as that of the majority class. Agrawal and Awekar (2018); Rathpisey and Adji (2019); Leonardelli et al. (2020) showed that replicating the samples randomly (random oversampling) from minority classes improved the performance of the hate speech classifiers. However, Vargas et al. (2021) argue that this method does not add any new information to the model and may also lead to the over-fitting of minority class samples. Rathpisey and Adji (2019) compared random oversampling with Synthetic Minority Oversampling TEchnique (SMOTE) Chawla et al., 2002 and Adaptive Synthetic (ADASYN) (He et al., 2008), methods that perform perturbations on feature vectors of the training set samples to create new samples. The authors showed that random oversampling outperformed these feature perturbation-based oversampling techniques. Similarly, Indurthi et al. (2019) applied SMOTE on deep learning-based sentence embeddings such as Universal Sentence Encoder (USE) (Cer et al., 2018) and InferSent (Conneau et al., 2017) to achieve class balance in the dataset. Prasad et al. (2021) showed that combining random oversampling of minority class and downsampling of majority class outperformed the individual oversampling or downsampling.

The class imbalance issue could also be addressed at the training step using class weighting. Class weighting gives higher weights to the minority class samples while computing loss (Banerjee et al., 2021; Rani et al., 2020).

Some researchers addressed the class imbalance problem at the classification metrics level by using the macro-average F1 measure (MacAvaney et al., 2019; Bosco et al.,

2018; Mandl et al., 2019). Macro-average F1 treats all the classes equally, whereas the micro-average or weighted-average F1 favors the F1 obtained on majority class samples (Sokolova and Lapalme, 2009). Thus, throughout this thesis, we use macro-average F1 measures as metrics to report hate speech classification performance.

2.10 Summary

In this chapter, we have presented an overview of a range of problems in hate speech classification and the state-of-the-art solutions used to mitigate them. We noticed that unclear definition of hate speech reduces the annotation quality as well as the generalizability of models across datasets. Biases in datasets are one of the factors leading to reduced model performance as well as reduced generalizability of the models trained on one dataset to perform well on other datasets. Hate speech datasets often have various biases, such as topic or author bias due to sampling strategy, and annotator bias due to factors such as the demography and the expertise of annotators.
Hate speech can be conveyed implicitly or explicitly. Classifying implicit hate speech characterized by the absence of profane words or the presence of sarcasm remains a challenging and is a problem rarely addressed. Hate speech also depends on the context such as a parent comment or the image associated with the comment. The absence of context information reduces the annotation quality as well as the model performance.
The evolving nature of hate speech over time, especially due to changes in the topic and vocabulary of hate is another challenging problem. This reduces the performance of the model trained on a particular data when tested on newer data. The other challenges in hate speech are the presence of out of vocabulary words, class imbalance, and few hate speech samples in the datasets.

3 Corpora and models for hate Speech classification

3.1 Hate speech classifiers

In this chapter, we present the advances in hate speech classification systems. Compared to Chapter 2, where we present the panorama of various challenges and solutions to the problems faced in hate speech classification, in this chapter, we focus on the advances in hate speech classifiers from pre-deep learning to the deep-learning era. We provide a detailed description of various datasets used for our experiments and the text preprocessing performed on them. We also present the different deep neural network-based models used in this thesis.

3.1.1 Pre-deep-learning approaches for hate speech classification

Initially, hate speech classification involved using various lexical features. The performance of a hate speech classification system depends on identifying the appropriate features and classification methods that use these features. The favorable features are the ones that capture the various complex phenomena of hate speech and improve model performance.
Razavi et al. (2010) used a dictionary of abusive words and hate words as a set of features for abusive language classification. However, hate speech can also be expressed without using any abusive words. In such cases, dictionary-based approaches can fail. In this case other language or hate speech relevant features can be helpful. Thus, Xu and Zhu (2010); Gitari et al. (2015) incorporated semantic relationships obtained using dependency parsers and Part-Of-Speech (POS) tagging along with the hate words dictionary to identify the words semantically related to hate words and performed rule-based hate speech detection. This method was extended by Chen et al. (2012), where the authors used a hate word dictionary and a dependency parser to identify the intensifiers in the sentence. The intensifiers were used to score the offensiveness of the input sentence, which was then used along with other features as input to other naïve Bayes and Support Vector Machines (SVMs) (Cortes and Vapnik, 1995) classifiers to improve hate speech classification.
Nobata et al. (2016) combined word and character N-gram-based features along with linguistic features such as length of comments, number of punctuations, number of uppercase characters, and syntactic features such as POS tags obtained from dependency parser for abusive language detection. The authors showed that character n-gram features were more robust than word n-gram features in the presence of misspelled words. Furthermore,

the authors showed that combining various features improved classification results. Fauzi and Yuniarti (2018) combined the advantages of various classifiers by using an ensemble approach, wherein Bag Of Words (BOW) features were used as input to various classifiers such as SVM, naïve Bayes, K-Nearest Neighbor, Maximum Entropy, and Random Forest and the decision for the final label was taken based on voting.

3.1.2 Deep learning-based approaches for hate speech classification

For classical machine learning techniques, selecting appropriate input features was domain-specific and was a complex task. Whereas, deep learning-based techniques performed an automatic selection of suitable input features, hence, requiring minimal domain-specific knowledge. Initially, DNNs were used for various image processing and speech processing applications (Pouyanfar et al., 2018). The advancement in deep learning techniques enabled them to be used for several Natural Language Processing (NLP) tasks.
Deep learning techniques outperformed classical machine learning methods and became state-of-the-art for hate speech classification. Badjatiya et al. (2017) performed hate speech classification using features such as character N-grams, Term Frequency-Inverse Document Frequency (TF-IDF), and BOW Vectors obtained using Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) embeddings as input to non-deep-learning-based classifiers such as SVMs, logistic regression, and Gradient Boosted Decision Trees (GBDTs). The results obtained by these classifiers were compared against using randomly initialized word-embeddings and pre-trained embeddings as input to deep-learning-based classifiers such as Convolutional Neural Networks (CNN) (Kim, 2014) and Long Short-Term Memory(LSTM) (Hochreiter and Schmidhuber, 1997). The study indicated that deep-learning-based classifiers outperformed classical machine learning-based methods. The advantage of using CNN is that it captures local patterns from the sequence of words of a given sentence. The Recurrent Neural Network (RNN) based models such as LSTMs or Gated Recurrent Unit (GRU) (Cho et al., 2014) capture the sequential information and long-range dependencies. Similarly, Del Vigna et al. (2017) compared SVM against LSTM with Word2Vec (Mikolov et al., 2013a,b) pre-trained embeddings as input features and observed that LSTM outperformed SVM for the Italian hate speech classification task. A similar study comparing classical machine learning models with deep learning-based models for hate speech classification was performed by Lee et al. (2018); Park and Fung (2017); Gambäck and Sikdar (2017). To consider the advantage of both the CNN and the RNN based models, Zhang et al. (2018) combined the CNN and GRU to form Convolutional-GRU (C-GRU), where the inputs are first passed through the convolutional layers, and the output of these layers is then used as input to the GRU layers. The results indicated that C-GRU performed better than the CNN model.
The performance of DNN based text classifiers depends on the input representation of the text used for the model. Pre-trained word embeddings are one such input representation used for various downstream NLP tasks. In the high dimensional vector space of the pre-trained embeddings, the semantically related words are close to each other. These representations are learned from huge unlabeled corpora. Pre-trained representations are commonly used as input representations for DNN based hate speech classification tasks

(Badjatiya et al., 2017; Gambäck and Sikdar, 2017; van Aken et al., 2018). The advantage of the pre-trained embeddings is that, when a word that doesn't occur in the training set appears in the test set, we still have a suitable pre-trained vector representation. Hence, usage of pre-trained embeddings partially solves the problem of out-of-vocabulary words in the test set.
The usage of pre-trained models like Embeddings from Language Models (ELMo)(Peters et al., 2018), Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), Robustly optimized BERT approach (RoBERTa) (Liu et al., 2019c), Generative Pre-Trained Transformer (GPT) (Radford et al., 2018), etc. proved to be state-of-the-art for various NLP tasks. These models are trained on a huge unlabeled corpus and can be easily fine-tuned to various downstream tasks using task-specific datasets.
Fine-tuning involves adding new task-specific layers to the model and updating the pre-trained model parameters along with learning new task-specific layers. Several works proposed to perform fine-tuning of these models for hate speech classification tasks. For example, Mozafari et al. (2019) fine-tuned BERT for the hate speech classification task. Similarly, Liu et al. (2019a) showed that fine-tuning BERT outperformed training an LSTM classifier. Banerjee et al. (2020) used an ensemble approach to combine the advantages of several transformer-based models like BERT, XLNeT (Yang et al., 2019), RoBERTa to improve hate speech classification results. A comparison of various DNN based embeddings and classifiers against fine-tuning of the pre-trained BERT model for hate speech detection is performed by Bodapati et al. (2019). The authors showed that fine-tuning BERT provides better classification results than using embeddings as input to the DNN based classifiers such as CNN.
Overall, there is a transition from using classical machine learning methods to DNN-based classifiers such as CNN, LSTM, BERT, etc. Hence, in this thesis, we use DNN and transformer-based models for hate speech classification.

To summarize, this chapter presents:

1. The different deep-learning-based models used in the thesis.
2. The hate speech datasets used in this thesis and the text pre-processing performed on them.

The rest of the chapter is organized as follows: We present the deep-learning-based models used in this thesis and the input representations in Section 3.2 and Section 3.3. The metrics used to evaluate the performance of the classifier is described in Section 3.4. Section 3.5 presents the various datasets used for our experiments. Section 3.6 describes the text preprocessing applied for these datasets.

3.2 Deep neural network models

In this section, we describe the different Deep Neural Network (DNN) layers, models, and activation functions used in this thesis.

3.2.1 Fully connected layer (or Dense layer)

A fully connected layer is the one whose neurons are connected to every neuron of the previous layer. They are most commonly used as the last few layers of complex DNN models, especially to process and compute the final output.

3.2.2 Multi-layer Perceptron (MLP)

It is one of the simplest forms of models, which consists of an input layer, one or several hidden layer(s), and an output layer. The dense layers are used for the hidden and output layer of the network. Generally, when more than one hidden layer is used, it can be called a DNN. We use this model as a classifier in Chapter 4.

3.2.3 Convolutional Neural Networks (CNN)

It consists of two main types of layers: convolutional layer and pooling layer.
Convolutional layer: The convolutional layers consist of several learnable filters. A filter has the same number of dimensions as that of input and is of smaller size (given a 2D image, 2D filters smaller than the width and height of the image is used for the convolution operation), they operate by convolving on the inputs to convert them to abstract feature maps, thus capturing the patterns of a given input.
Pooling layer: The pooling layers are used to reduce the dimension of data, they are used to convert the output of several neurons to a single one.
CNN was traditionally used in the domain of image processing and is effective at capturing patterns. Originally intended to operate on 2D space, it involves using a layer with convolving filters applied to the local features (LeCun et al., 1989). Kim (2014) demonstrated the efficient use of CNN for text classification on various benchmark datasets.

3.2.4 Gated Recurrent Unit (GRU)

A GRU is a Recurrent Neural Network (RNN) based model, used for sequential processing of the data. Text data can be treated as a sequence of words. Thus, GRU can be effectively used to capture these sequential patterns and relationships among words. Compared to the standard RNN, GRUs overcomes the vanishing gradient problem and capture long-range dependencies.
Figure 3.1 shows the architecture of a GRU unit. In the Figure, x_t is the input vector, h_t is the output hidden vector, h_{t-1} is the output hidden vector from the previous time step, r_t represents the reset gate vector, z_t represents the update gate vector, $\hat{h}_t$ is the activation vector, σ is the sigmoid activation, $tanh$ is the hyperbolic tangent activation, $\times$ indicate element-wise multiplication of vectors.
The output hidden vector h_t can be computed as below:

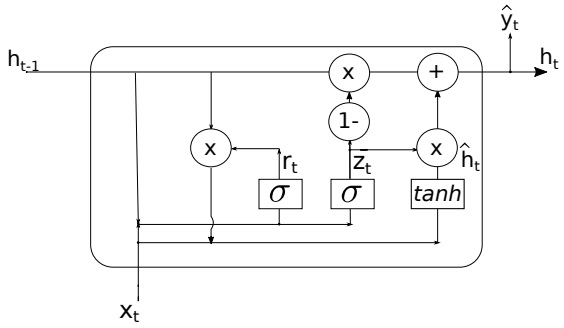


Figure 3.1: Architecture of the GRU unit.

$$
\begin{aligned}
z_t &= \sigma(W_z * x_t + U_z * h_{t-1} + b_z) \\
r_t &= \sigma(W_r * x_t + U_r * h_{t-1} + b_r) \\
\hat{h_t} &= tanh(W_h * x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h_t}
\end{aligned}
$$

Where $\odot$ is the element-wise multiplication. W, U, b are learnable parameters.

3.2.5 Convolutional-Gated Recurrent Unit (C-GRU)

To leverage the advantages of both CNN and RNN based models, Zhang et al. (2018) combined convolutional layers with RNN based GRU layers for hate speech classification and showed improved classification results compared to the CNN classifier. Thus, in Chapter 5, we explore C-GRU as a DNN based classifier for hate speech classification. The schema of the C-GRU model is shown in Figure 3.2.

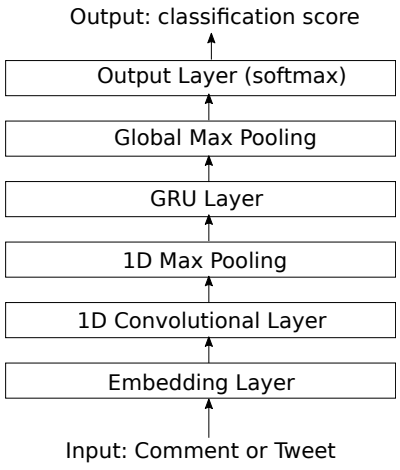


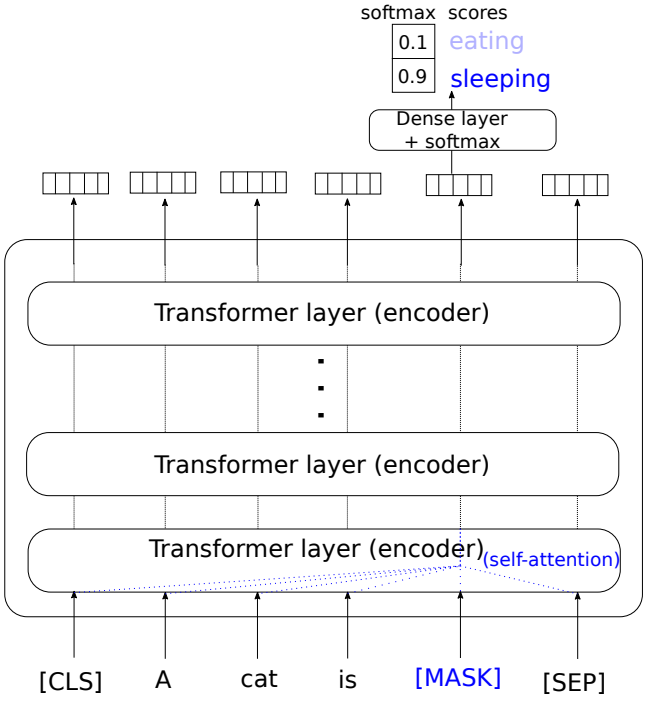
Figure 3.2: Schema for C-GRU classifier.

3.2.6 Transformer layers

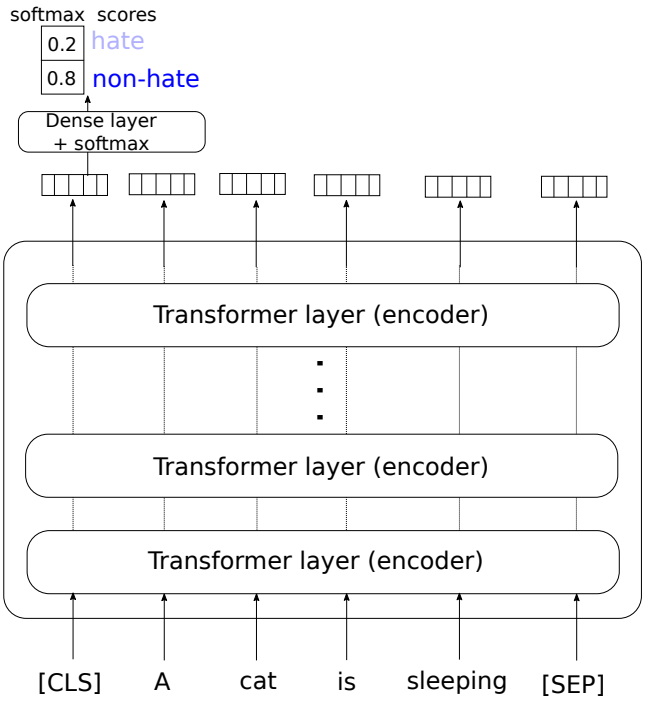
Transformer layers are building blocks of transformer-based models (Vaswani et al., 2017). They process a sequence of text without time-step-based processing (processing one word or token at a time) like in RNN based models. They process an entire sequence at once, thus achieving parallelization. Transformers use the self-attention mechanism, a method to capture the dependencies of a given token with other tokens in the sequence.
Each transformer layer consists of two sub-layers, a multi-head attention layer and a feed-forward neural network layer (dense layer). The multi-head attention allows the model to jointly attend representation from different subspaces, thus capturing different relationships between words or tokens. The feed-forward network is used to apply linear transformations to the outputs of the attention layer.
The transformer model was originally used for the translation tasks. The model architecture consists of two blocks: the encoder and the decoder. Both encoder and decoder blocks use transformer layers. The encoder is used to estimate the representations of the input text. The representations learned by the encoder are used by the decoder to generate the translated sequence.

3.2.7 Bidirectional Encoder Representations from Transformers (BERT)

BERT (Devlin et al., 2019) is a transformer-based auto-encoder model. BERT uses only the encoder block of the transformer architecture. The transformer layers of the BERT model use the *self-attention* mechanism. The self-attention mechanism is a way to capture the context information from both the directions (left and right) of a given token. The model is pre-trained using the *masked language model* and the *next sentence prediction* objective.
The masked language model is a self-supervised learning method, which involves corrupting a few input tokens by replacing them with the '[MASK]' token and predicting the masked tokens at the output of the model. The next sentence prediction involves using two sentences A and B together as the input to the model and predicting if the sentence B follows the sentence A in the original document. The next sentence prediction task was used to capture the relationship between two sentences, which can be useful for tasks such as 'question-answering' and 'natural language inference'. Figure 3.3a illustrates pre-training of the BERT model using the masked language modeling objective.
The BERT model uses word-piece tokens as input to the model. Word-piece tokenization uses a fixed-sized vocabulary containing words and sub-word tokens. Word piece tokenization splits a given word to sub-words if the word is not found in the vocabulary. For example, if the word "singing" is not found in the vocabulary, it may split into two word-piece tokens "sing" and "##ing". The input tokens are prepended with the '[CLS]' token and appended with the '[SEP]' token. The '[CLS]' token is considered as a special token for the classification task. Due to the self-attention mechanism, the output of the '[CLS]' token can capture a representation of the entire input sentence. Fine-tuning of the BERT model for classification involves adding a new classification layer (dense layer) on top of the BERT model and updating the pre-trained BERT model parameters along



(a) Illustration of the BERT pre-training using *masked language model* objective.



(b) Illustration of the BERT fine-tuning for classification task.

Figure 3.3: Illustration of the BERT model pre-training and fine-tuning.

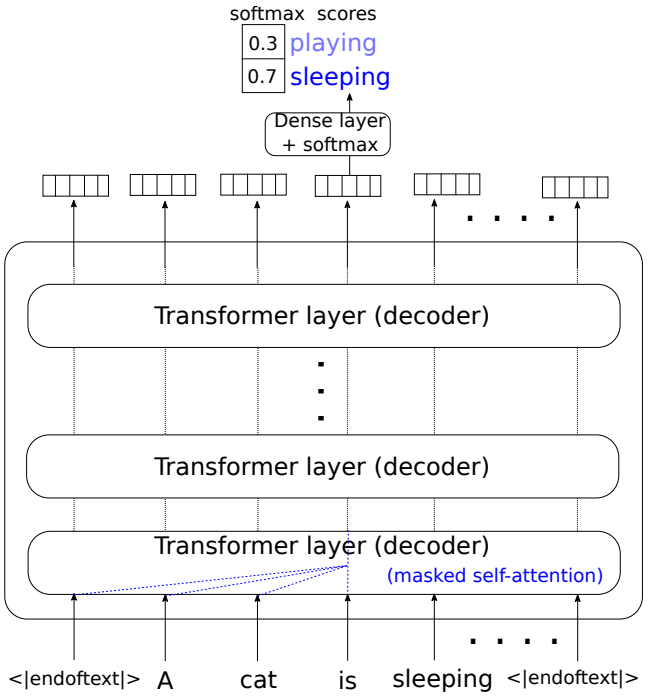
with the parameters of the classification layer, using the task dataset. The hidden state output of the last hidden layer corresponding to the '[CLS]' token is used as an input to the classification layer. Figure 3.3b illustrates fine-tuning of the BERT model for the classification task.
For our experiments, we use the pre-trained 'bert-base-uncased' model. It has a stack of 12 transformer encoder layers with 110M trainable parameters, and the embedding dimension is 768. The model is pre-trained on BooksCorpus (Zhu et al., 2015) and English Wikipedia, containing 800 million words and 2500 million words, respectively. We use the implementation of Huggingface's transformers API (Wolf et al., 2020) for our implementation.

3.2.8 Generative Pre-Trained Transformer-2 (GPT-2)

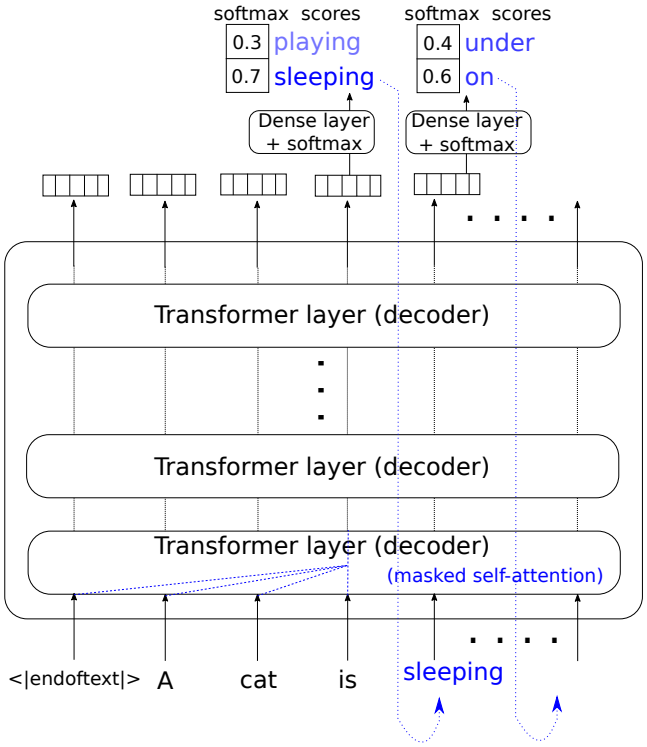
The GPT-2 (Radford et al., 2019) model is an auto-regressive transformer-based model. GPT-2 uses only the decoder block of the transformer architecture. It is pre-trained with the objective of *generative pre-training*. Generative pre-training is an self-supervised learning method and is similar to the objective of language modeling (explained in Section 5.2.1). This objective involves predicting the next token based on the given input context (previous tokens). Unlike BERT, which uses self-attention in the bi-directional context (left and right) for predicting the masked tokens, GPT-2 model uses *masked self-attention* (or uni-directional self-attention) to predict a token based on the left context. Masked self-attention uses only the left context by masking the attention scores for all the tokens on the right.
Similar to the BERT model, the pre-trained GPT-2 model can be fine-tuned on various downstream tasks. However, due to its training objective, it is more suitable for text generation tasks. Similar to the BERT model, GPT-2 uses word-piece tokens as input to the model. The input tokens are prepended and appended with the '<|endoftext|>' token to indicate the start and end of the text or the sentence. In Chapter 5, we use the GPT-2 model to generate synthetic samples for data augmentation. Figure 3.4a illustrates the pre-training of the GPT-2 model using the generative pre-training objective and masked self-attention.
To generate a sequence, the GPT-2 model can be provided with context tokens (prompt text). The model uses these context tokens to predict the next token. The generated token is then used along with the original context tokens to generate a new token, and the process continues. Figure 3.4b illustrates the text generation using the GPT-2 model. For our experiments, we use 'GPT-2 large', it has 24 transformer decoder layers containing 774M model parameters. This model is pre-trained on 40 GB of text obtained from various web sources.

3.2.9 Rectified Linear Unit (ReLU) activation

In the DNN layers, the output of a neuron is a linear combination of its inputs. However, a non-linear activation on the output of neurons transforms these linear results into non-linear results, allowing to solve some non-trivial problems and also reducing the number



(a) Illustration of the GPT-2 model pre-training using *masked self-attention*.



(b) Illustration of sequence generation using the GPT-2 model.

Figure 3.4: Illustration of the GPT-2 model pre-training and sequence generation.

of required neurons. A ReLU is a non-linear activation function r is defined as:

$$r(x) = max(0, x)$$

3.2.10 Softmax activation

It is often used as an activation function on the output layer of the neural network for the multi-class classification task. Softmax normalizes the outputs to the probability distribution over predicted classes. The softmax function S over n output units corresponding to n classes for the class i is computed as below:

$$S(y)_i = \frac{exp(y_i)}{\sum_{j=1}^{n} exp(y_j)}$$

The sum of all the components of $S(y)_i$ is 1.

3.3 Input Representation for Hate Speech Classifiers

3.3.1 Word-embeddings

The classification models require a suitable numerical representation of the text as input. The words in a given text sequence can be represented in numerical form using one-hot vectors. The one-hot vector is a high-dimensional representation, having the dimension equal to the number of words in the vocabulary. A word in the vocabulary is represented using this vector with all the values equal to zero, except one, indicating the position of the word in the vocabulary. However, as the length of the one-hot vector increases with the increase in the number of words in the vocabulary, a table formed with one-hot vectors quadratically increases with the size of the vocabulary. Furthermore, this one-hot vector representation is discreet. A continuous representation could be more powerful and could allow mathematical operations. To perform this, a projection of words in the continuous space can be accomplished. Thus word-embeddings are a numerical representation of a word using fixed-length vectors. Words that are closer in the vector space are expected to be related. These embedding vectors can be learned using neural networks, factorization of the co-occurrence matrix (Li et al., 2015), or explicit representation using positive pointwise mutual information (Levy and Goldberg, 2014), etc. Using neural networks, these vectors can also be randomly initialized and the values of these vectors can be learned during the training of the model. In Chapter 5, we use word-embeddings learned from the available corpus as input to the C-GRU model for hate speech classification.

3.3.2 Pre-trained word-embeddings

Pre-trained embeddings are numerical representations using fixed-length vectors, learned from a huge unlabeled corpus. By using pre-trained embeddings we leverage the strong representational capabilities of these embeddings learned from the huge unlabeled corpus. Some of the commonly used word embeddings are Word2Vec (Mikolov et al., 2013a,b)

and fastText (Bojanowski et al., 2016; Mikolov et al., 2018). These pre-trained word embeddings can also be used to initialize the embeddings layer and can be further updated by training the model on the downstream tasks.

Word2Vec: These embeddings are trained using the Continuous Bag of Words (CBOW) and the Continuous Skip-gram algorithms. To learn the embeddings of a given word, the algorithms use a small window of neighboring words (context). Thus, the final representation of these embeddings depends on the context in which the words frequently occur.

fastText: It is an extension of Word2Vec embeddings. These embeddings are learned using the CBOW based model. Unlike Word2Vec, this fastText incorporated bag of character N-grams (Bojanowski et al., 2016) to obtain the embedding of the words in addition to learning from the CBOW algorithm. Wherein, the final embedding representation is obtained by averaging the word embedding obtained from the CBOW algorithms, and a bag of character N-grams model.

BERT embeddings: The pre-trained BERT model can be used to obtain contextual embeddings. Because of the self-attention mechanism, the embeddings for a given token provided by the BERT model depend on left and right context. Unlike the static word embeddings such as Word2Vec or fastText, the embeddings provided by the BERT model are dynamic. In static word embeddings, a given word will have the same vector representation when it occurs in two different contexts. However, for contextual embeddings, the vector representation depends on the context in which the word occurs. For example, the word "playing" will have one BERT embedding when it occurs in the context "He likes playing guitar", and a different BERT embedding when it occurs in "He likes playing football".

3.3.3 Pre-trained sentence-embeddings

Similar to word embeddings that provide numerical representation for words, sentence embeddings provide numerical representations for the entire sentence. Thus, sentence embeddings are fixed-length vector representations for a sentence, where the semantically related sentences are close to each other in their embedding space. Universal Sentence Encoder (USE) (Cer et al., 2018), InferSent (Conneau et al., 2017) are few among the known sentence embeddings. We use pre-trained USE embeddings as sentence representations for MLP classifier in Chapter 4.

Universal Sentence Encoder (USE) embedding: The authors of USE explored two architectures: transformer and Deep Averaging Network (DAN) (Vaswani et al., 2017). The transformer model provides sentence embeddings having higher accuracy, but has higher computational cost than the DAN model. To obtain the final sentence embeddings, the output embeddings obtained for all the tokens from the model are averaged. The models are trained on the multi-task learning objective, where the sentence embeddings are used as the input to various task-specific layers. The tasks for multi-task learning include classification, conversational input-response (Henderson et al., 2017), skip-thought (Kiros et al., 2015), etc. The model is trained using Wikipedia, web crawled

data, and Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015).

3.4 Classification metrics

In this thesis, we use percentage macro-F1 as the classification metric to report the model performance. Macro-average F1 treats all the classes equally, whereas the micro-average (accuracy) or weighted-average F1 favors the F1 obtained on majority class samples (Sokolova and Lapalme, 2009).
The value indicated by macro-average F1 ranges between 0 and 1, where 1 indicates the best performance. In this thesis, we multiply macro-average F1 by 100 to obtain percentage macro-average F1 measure. F1-measure is calculated as follow:

$$F1 = \frac{2 * (precision * recall)}{(precision + recall)}$$

where, precision is the ratio between number of samples correctly predicted as particular class A and total number of samples predicted as class A by the classifier; recall is the ratio between number of samples correctly predicted as class A and total number of samples that should have been predicted as class A.
Macro-average F1-measure provides the arithmetic mean of F1-measures of all classes:

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

where, C is the total number of classes.

3.5 Hate speech datasets

We consider five widely used hate speech datasets, four are from Twitter, namely, 'Davidson' (Davidson et al., 2017), 'Founta' (Founta et al., 2018), 'Hateval' (Basile et al., 2019), and 'Waseem' (Waseem and Hovy, 2016). The fifth dataset is sampled from Wikipedia talk pages (Wulczyn et al., 2017), referred as 'Wikipedia'. The statistics for these datasets are provided in Table 3.1.

3.5.1 Davidson

This dataset was collected by sampling the tweets based on the keywords from the hatebase lexicon.[1] The dataset is annotated into three classes: 'neither', 'offensive language', and 'hate speech'. Further in this thesis, we refer to these classes as 'normal', 'abusive', and 'hateful', respectively. The Davidson dataset uses the following definition for the 'Hateful' class *"language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group"* (Davidson et al., 2017). The dataset has about 25K tweets and is annotated by crowdsourcing.

[1] https://www.hatebase.org

Dataset	Total	Class labels			average number of words per Tweet or Comment
		normal	abusive	hateful	
Davidson	24.8K	4.2K (16.9%)	19.2K (77.4%)	1.4K (5.6%)	14.1
Founta	86.0K	53.8K (62.6%)	27.2K (31.6%)	5.0K (5.8%)	16.9
		non-hateful		hateful	
Hateval	13.0K	7.5K (57.7%)		5.5K (42.3%)	21.6
Waseem	10.9K	8.0K (73.4%)		2.9K (26.6%)	14.7
		non-toxic		toxic	
Wikipedia	159.7K	131.7K (82.5%)		28.0K (17.5%)	66.9

Table 3.1: Statistics of the number of samples and average number of words per Tweet or Comment in Davidson, Founta, Hateval, Waseem, and Wikipedia datasets.

At least three annotators annotated a tweet, and the dataset has an inter-annotator agreement of 92%.

3.5.2 Founta

The Founta dataset has four classes, namely, 'normal', 'abusive', 'hateful', and 'spam'. We have discarded the samples belonging to the 'spam' class, for all the experiments performed on this dataset. The removal of samples from the 'spam' class reduced the size of this dataset from 100K tweets to 86.9K tweets. A large part of this dataset is collected using random sampling. Since hate and abusive speech occurs in a very small percentage, the authors chose to perform boosted sampling. The boosted sampling involves pre-selecting the tweets for annotation based on sentiment analysis, wherein samples having negative sentiment are selected and added into randomly sampled data. Founta dataset uses the same definition of hate speech as that of the Davidson dataset. Each comment is annotated by at least 5 annotators using crowdsourcing.

3.5.3 Hateval

This dataset was published and used for the 'SemEval-2019' shared task, to identify hate speech against women and immigrants. The dataset contains tweets sampled from Twitter. The shared task involves two datasets, one for the Spanish language, and the other for the English language. For our study, we have used the English dataset. The dataset is annotated into two classes, namely, 'hateful' and 'non-hateful'. Because of the very poor classification performance of this dataset, with the mean macro-average F1 score

obtained by the participants being 44.84, we have considered this dataset only to train our multi-corpus learning model in Chapter 6, and have not used it for experiments in other chapters. The dataset provides 9K, 3K, and 1K samples for training, development, and test sets, respectively.

3.5.4 Waseem

This dataset was sampled from Twitter using the keywords containing racial and sexual slurs. This dataset has three classes: 'racism', 'sexism', and 'none'. A total of 16.9K tweets, with 3.4K, 2.0K, and 11.6K for 'sexism', 'racism', and 'none', respectively. The dataset is annotated by two annotators (Waseem et al., 2018) and has an inter-annotator agreement score of 84%. The authors of this dataset share only the Tweet-id's of the tweets and their class labels. However, due to the filtering strategy of Twitter to remove the hateful contents, after retrieval of data using the Tweet-id's our dataset has only 20 samples for the 'racism' class, 2.9K samples for the 'sexism' class, and 8.0K samples for 'none' class respectively. A similar number of samples due to the missing tweets is reported by (Bodapati et al., 2019). In Table 3.1, we refer to the 'sexism' class as 'hateful', and the 'none' class as 'non-hateful'. We discard the samples from the 'racism' class due to a very few number of samples.

3.5.5 Wikipedia

The dataset was collected in the framework of the *Wikipedia Detox* project. It contains the comments from the user talks pages. The dataset is annotated for 'personal attacks', 'aggression', and 'toxicity'. We use the 'toxicity' part of the dataset, annotated into five labels, 'very toxic', 'toxic', 'neither', 'healthy', and 'very healthy'. The dataset has a total of 159.7K comments, annotated by crowdsourcing, and each comment is annotated by approximately ten annotators. For many comments in this corpus, there is a disagreement between annotators. Sometimes, it is difficult to define a dominant label for a comment. For example, a comment having only one annotation of 'toxic' or 'very toxic', and nine annotations for 'neutral' cannot be easily considered as 'toxic'. Thus, to perform the binary classification ('toxic' versus 'non-toxic'), for each comment, we decided to use the following labeling scheme:

```
if [(# of very toxic and toxic annotations) >
   (# of healthy and very healthy annotations)]
   and [(# of very toxic and toxic annotations) > 2]
    comment is toxic
otherwise
    comment is non-toxic
```

The dataset provides 95.7K, 32.1K, and 31.9K samples for training, development, and test set, respectively.

The labeling schemes of Wikipedia differ from that of Hateval and Wassem. The samples

with abusive or hateful language together belong to the 'toxic' class in Wikipedia dataset. However, abusive language samples belong to the 'non-hateful' class, and hateful samples belong to 'hateful' class for Hateval and Waseem datasets.

3.6 Input preprocessing

In this section, we first describe the sets of text preprocessing specific for Twitter and Wikipedia datasets. We then describe the text processing that is common for both types of datasets.

3.6.1 Twitter specific preprocessing

We apply this preprocessing for the four Twitter corpora: Davidson, Founta, Hateval, and Waseem datasets. Twitter user handles beginning with the '@' symbol are changed to '@USER'. The '#' symbol in the hashtag is removed, and the multi-word hashtags are split based on the presence of the uppercase characters in the hashtags. For example, '#leaveThisPlace' is changed to 'leave This Place'. The multi-word hashtags without any uppercase characters are unchanged. The word 'RT' indicating re-tweet is removed.

3.6.2 Wikipedia specific preprocessing

We apply this preprocessing only for the Wikipedia dataset. We remove the occurrences of the text 'NEWLINE_TOKEN' which occurs very frequently in most of the comments and indicates the presence of a new line in a given comment.

3.6.3 Preprocessing common to all datasets

We removed the numbers, newline, and special characters except '.', ',', '!', '?', and *apostrophe*. Repeated occurrences of the same special character are reduced to a single one. All the URLs and emoticons are also removed. Finally, all the data is lower-cased.

3.7 Summary

In this chapter, we have presented the literature on pre-deep learning and deep-learning approaches for hate speech classification. Deep Neural Network (DNN) based classifiers have been shown to out-perform classical machine learning-based classifiers for hate speech classification. Thus, in this thesis, we have explored DNN-based approaches for classification. Various DNN based models used in this thesis are presented in this chapter. We have described the models such as multi-layer perceptron, GRU, and C-GRU. The transformer-based models (BERT and GPT-2) and their pre-training and fine-tuning techniques are also described.
We have also described pre-trained word embeddings and sentence embeddings. Embeddings are a vector representation of a word or a sentence, that is commonly used as input representations to DNN based classifiers. The pre-trained embeddings are learned from

huge corpora. In these pre-trained representations, the related words or sentences are close to each other in the vector space. Thus, they are suitable for various downstream natural language processing tasks.
We have described macro-averaged F1, which is the classification metrics used in this thesis. We have also presented various hate speech datasets. Wherein, we have considered four datasets from Twitter (Davidson, Founta, Hateval, and Waseem) and one from Wikipedia. Wikipedia allows the two-class classification in toxic and non-toxic speech. Hateval and Waseem allow the two-class classification in hate and non-hate speech. Whereas, Davidson and Founta datasets are used to perform the multi-class classification of hate speech into normal, abusive, and hate speech, respectively. This allows us to study the complex phenomena of hate speech and its overlap with other classes. We have also described the text preprocessing used on these datasets.

4 Semi-supervised learning for hate speech classification

4.1 Introduction

In this chapter, the semi-supervised learning approach for the hate speech classification is introduced. The performance of the deep-learning-based classifier depends on the amount of available labeled training data. In many real-life scenarios, there is a limited amount of labeled data and abundant quantity of unlabeled data. In this chapter, we aim to study the semi-supervised learning approach to take advantage of the unlabeled data along with the labeled data to train the hate speech classifier.
To improve a classification model by increasing the amount of available training data. We present a detailed study of data augmentation techniques in Natural Language Processing (NLP) and hate speech in Section 5.1.1. Although the data augmentation techniques have shown improvement in hate speech classification performance, they use only the labeled training data and fail to take advantage of the available unlabeled data. However, semi-supervised learning can be used to combine the unlabeled data with the labeled data to improve model performance.

4.1.1 Semi-supervised learning in natural language processing

Semi-supervised learning is a technique to combine a small amount of labeled data with a large amount of unlabeled data during training (Abney, 2007) to improve the performance of the classifiers. Various semi-supervised learning approaches are explored in the field of NLP. A group of semi-supervised learning methods uses bootstrapping techniques such as self-training (Yarowsky, 1995), where a model is trained with a small amount of labeled data, and the trained model is then used to label the unlabeled data. The unlabeled samples with high prediction scores are added to the training set, and the process is repeated. This has been used for various NLP tasks such as classification (Wang et al., 2008), Part-Of-Speech (POS) tagging (Huang et al., 2009), word-sense disambiguation (Yarowsky, 1995; Mihalcea, 2004), etc. Blum and Mitchell (1998) extended self-training to co-training, where two models are trained with an independent set of features, and the samples from the unlabeled set with confident predictions from both models are iteratively added to the training set. Similarly, tri-training (Zhou and Li, 2005) extends this method by using three models, where an unlabeled sample is labeled and added to the training set of the first model if the other two models predict the same label. Søgaard (2010) improved tri-training to tri-training with disagreement, where, an unlabeled sample is labeled and added to the training set of the first model if the

other two models predict the same label and the first model predicts a different label. Thus, strengthening each model on its weak points. The parameters of the three models used for tri-training were shared using multi-task learning to improve the performance of tri-training by Ruder and Plank (2018). The performance of self-training approaches for semi-supervised learning depends on the strength of the initial model trained with a few amount of labeled samples. Similarly, the performance of a Deep Neural Network (DNN) based classifier depends on the amount of labeled data. Thus using self-training may not be ideal for DNN based hate speech classifier when the number of labeled training samples is too small to obtain a well-trained initial model.

Another group of approaches annotates the unlabeled text data using an external knowledge-base containing auxiliary information or uses hand-crafted rules to label the unlabeled data. For example, the list of entries from the *'Persons'* and the *'Organizations'* categories of Wikipedia was used to label the data for the Named Entity Recognition (NER) task (Dembowski et al., 2019). Similarly, Awasthi et al. (2019) used hand-crafted rules, where the presence of certain keywords or phrases was used to label the data for slot-filling and classification tasks. However, such rule-based annotated data is unreliable as it contains noise, thus requiring noise handling mechanisms (Paul et al., 2019; Hedderich and Klakow, 2018). Furthermore, hate speech can be conveyed through sarcasm or irony, and hence, nay not contain hate words. This makes keywords or rule-based approaches for labeling hate speech data inefficient (MacAvaney et al., 2019).

The last group of approaches relies on constructing graphs to build the connectivity between the labeled and the unlabeled data, then transferring the labels from the labeled data to the connected unlabeled data (Xiaojin and Zoubin, 2002; Zhou et al., 2004; Belkin et al., 2006). The data samples are represented as vertices of the graph and are connected using edges. Graph-based semi-supervised learning have been successfully employed for various NLP tasks such as classification (Widmann and Verberne, 2017; Mao et al., 2011; Alexandrescu and Kirchhoff, 2007), POS tagging (Subramanya et al., 2010), etc.

4.1.2 Semi-supervised learning in hate speech classification

Rosenthal et al. (2021) used a co-training based approach called democratic co-training (Zhou and Goldman, 2004) for semi-supervised learning to create a larger dataset to improve offensive language classification. Democratic co-training involves different type of classification algorithms to label the unlabeled data: the unlabeled samples obtaining same labels with higher model confidence from majority number of classifiers are selected and labeled. These samples are added to the training set and the classifiers predicting incorrect labels are retrained. Bansal et al. (2021) showed small improvements for toxic-span detection using self-training-based semi-supervised learning. A self-training-based approach was used to enhance fine-grained sexism detection by Abburi et al. (2021). A combination of rule-based approach using a list of keywords and a self-training-based approach was employed by Khatri et al. (2018) to perform offensive content detection in a chatbot application. Thus, with very few works employing semi-supervised learning for the hate speech classification task, the usage of unlabeled data has been little explored. Therefore, we decided to explore label propagation (Xiaojin and Zoubin, 2002) based

semi-supervised learning for the hate speech classification task.
Label propagation is a graph-based semi-supervision technique analogous to the k-Nearest-Neighbours algorithm. It assumes that data points close to each other tend to have a similar label. This algorithm relies on the representations of data points to create a distance graph that captures the proximity of the samples represented as vertices of the graph.
Recently, pre-trained word embeddings such as Word2Vec (Mikolov et al., 2013a,b), fastText (Bojanowski et al., 2016), Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) have been used for representing words for hate speech classification (Waseem et al., 2017a; Badjatiya et al., 2017). Furthermore, pre-trained sentence embeddings such as InferSent (Conneau et al., 2017), Universal Sentence Encoder(Cer et al., 2018), Embeddings from Language Models (ELMo) (Peters et al., 2018) have been used for the task of hate speech classification (Indurthi et al., 2019; Bojkovskỳ and Pikuliak, 2019). Compared to the word embeddings that provide high dimensional numeric representations for each word, the sentence embeddings provide representations for the entire sentence. The sentences that are semantically similar are close to each other in the embedding space. Our work aims to show that these pre-trained sentence embeddings are generic, hence, we transform the pre-trained sentence embeddings to task-specific representations that can be helpful for label propagation (D'Sa et al., 2020).

The contributions of this chapter are:

1. The evaluation of label propagation-based semi-supervised learning for hate speech classification.
2. The comparison of pre-trained representations with task-specific representations learned from a small labeled corpus used for the label propagation algorithm.
3. The evaluation of label propagation-based semi-supervised learning for hate speech classification with unlabeled data having different distribution than the labeled data.

The rest of the chapter is organized as follows: Section 4.2 describes the label propagation algorithm and our proposed methodology of label propagation-based semi-supervised training of hate speech classification. Section 4.3 details the considered datasets and model parameters used for our experiments. We present our results of semi-supervised training and analyze the results in Section 4.4 and conclude in Section 4.5.

4.2 Semi-supervised learning using label propagation and sentence embeddings

In this section, we briefly describe the label propagation algorithm and sentence embeddings. This is followed by our methodology for semi-supervised training for hate speech classification using label propagation and a Multi-Layer Perceptron (MLP).

4.2.1 Label propagation

Label propagation is a graph-based semi-supervised learning technique that uses the labels from the labeled data to transduce the labels to the unlabeled data. Label propagation considers two sets: $\{(x_1, y_1) \dots (x_l, y_l)\} \in L$ and $\{(x_{l+1}) \dots (x_{l+u})\} \in U$, where L is a labeled set and U is an unlabeled set, each containing l and u samples respectively, with assumption that $l << u$. $\{x_1 \dots x_l, x_{l+1} \dots x_{l+u}\}$ are the feature representations of the samples in L and U and $\{y_1 \dots y_l\}$ are the labels corresponding to samples $\{x_1 \dots x_l\}$ in labeled set L. Here, $\{y_1 \dots y_l\} \in \{1 \dots C\}$, where C is the number of classes, and $\{x_1 \dots x_{l+u}\} \in \mathbb{R}^Z$, where Z is the dimension of the feature vector space. The labeled set L consists of samples belonging to all the classes. The objective of label propagation is to estimate the labels $\{y_{l+1} \dots y_{l+u}\}$ for the samples in unlabeled set U using $\{x_1 \dots x_l, x_{l+1} \dots x_{l+u}\}$ and $\{y_1 \dots y_l\}$.
The algorithm constructs a graph $G = (V, E)$, where V is the set of vertices representing samples in set L and U, and the edges in set E represents the similarity between two nodes i and j with weight w_{ij}. The weight w_{ij} is computed such that nodes with smaller distances (similar nodes) will have larger weights. In case of Euclidean distance, w_{ij} can be computed as below:

$$w_{ij} = exp\left(-\frac{d_{ij}}{\sigma^2}\right) = exp\left(-\frac{\sum_{z=1}^{Z}(x_i^z - x_j^z)^2}{\sigma^2}\right)$$

Where, Z is the dimension of the feature representations for samples in L and U, and σ is a hyper-parameter for scaling the effect of label propagation.
The algorithm uses a $(l+u) \times (l+u)$ probabilistic transition matrix T, where the element T_{ij} indicates the probability of transiting from the node i to the node j (indicated by $P(i \to j)$), obtained as below:

$$T_{ij} = P(i \to j) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}}$$

The algorithm defines a $(l+u) \times C$ label matrix Y, containing the probabilities of labels for all the samples in the labeled and unlabeled set. It iteratively updates the labels in Y, by without changing the labels of the labeled set, and until Y converges:

$$\hat{Y} \leftarrow TY$$

4.2.2 Sentence embeddings as feature representations

The label propagation algorithm works on the assumption that samples similar to each in some representations other are close to each other in the feature vector space. Thus, choosing the appropriate feature space becomes an important factor to transfer the reliable labels from the labeled set to the unlabeled set. Sentence embeddings are a form of high dimensional, fixed-length vector representations that captures the semantics of the

sentence. Usually, these embeddings are learned from large unlabeled corpora. Semantically similar sentences are close to each other in this vector space, therefore, could be used as feature representations for various downstream tasks. In our study, we use the sentence embeddings obtained from the pre-trained transformer (Vaswani et al., 2017) based Universal Sentence Encoder (USE) (Cer et al., 2018) to represent the tweets.

4.2.3 Proposed methodology

Figure 4.1 outlines the semi-supervised learning setup adopted in our study. We use a Multilayer Perceptron (MLP) for two purposes: (a) to transform pre-trained representations to task-specific representations; (b) to perform binary or multi-class classification.

Representation transformation: Our approach involves transforming the pre-trained representations (USE embeddings) to task-specific representation. We do this because the pre-trained representations are generic and can fail to capture the downstream class label information, especially for the hate speech classification task. To transform the pre-trained representations to task-specific representations, we first train an MLP classifier using the labeled set L. We use USE embeddings as input to this model. The MLP consists of two hidden layers, the activation obtained from the first or the second hidden layer is considered as the transformed task-specific representations ('h1 representations' and 'h2 representations' in Figure 4.1).

Labeling the unlabeled data: After training the MLP classifier, we pass the pre-trained representations for samples from the labeled set and the unlabeled set as input to the trained MLP classifier, to obtain the task-specific representations. These task-specific representations are used as input to the label propagation algorithm.
Label propagation algorithm uses the task-specific representations of labeled and unlabeled data, along with the labels for the labeled data. This algorithm provides labels for the samples in the unlabeled set.

Classifier training: We combine the labeled set and the unlabeled set to train the MLP classifier. We use the pre-trained embeddings (USE embeddings) of the tweets as inputs for our MLP classifier. We use the original labels for the samples in the labeled set and the labels obtained from label propagation for the unlabeled set.

4.3 Experimental setup

4.3.1 Data description

We consider two datasets, dataset by Founta et al. (2018) and Davidson et al. (2017), containing tweets sampled from Twitter. Further in this chapter, we refer to them as 'Founta' and 'Davidson', respectively. A detailed description of the considered datasets is provided in Section 3.5. For the fine-grained classification of hate speech, we use three

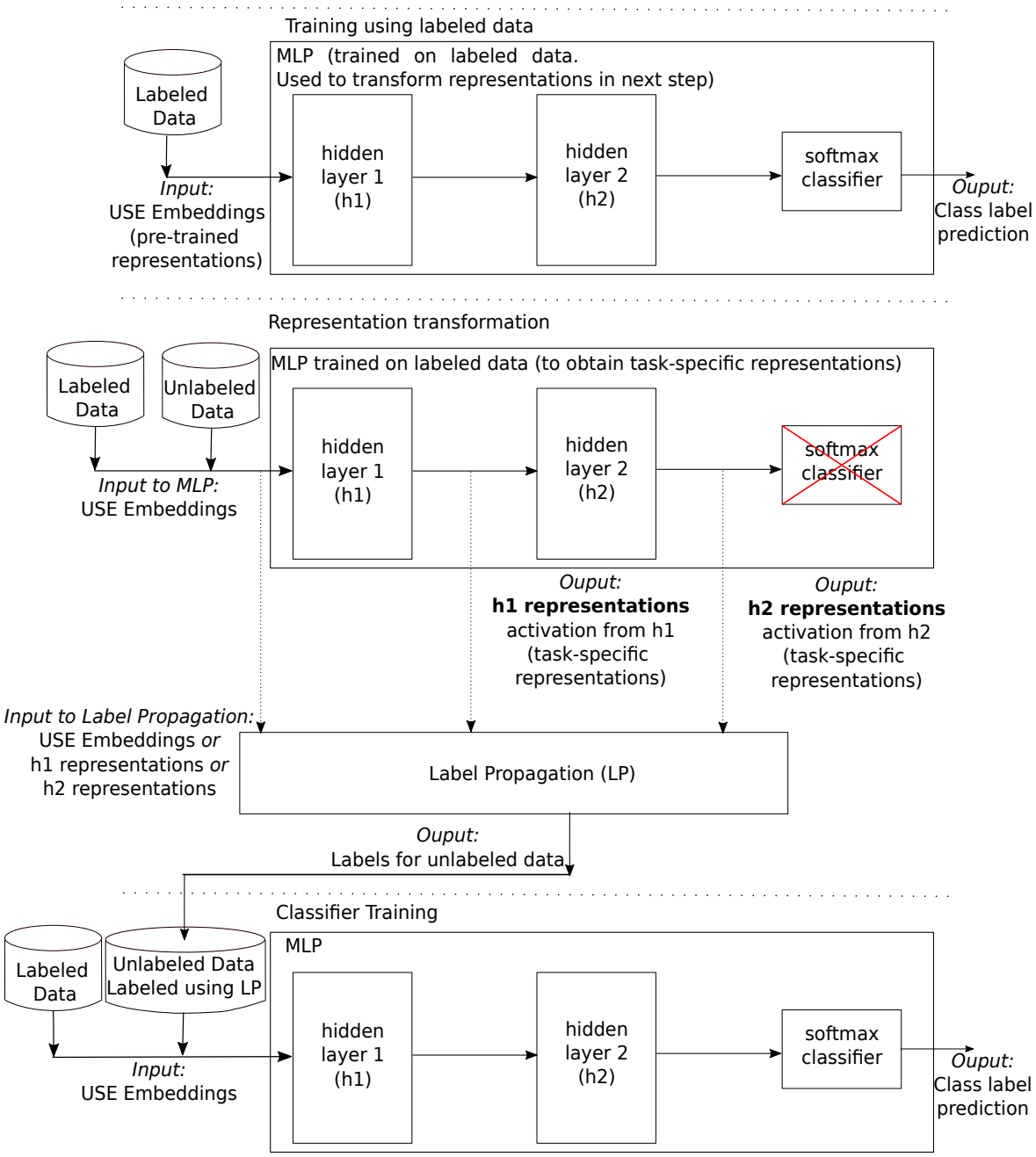


Figure 4.1: Block diagram for semi-supervised learning.

class labels given by datasets. However, for binary classification, we combine the samples for 'abusive' and 'hateful' classes under the single umbrella term 'toxic speech'. For both these datasets, we have applied the text processing described in Section 3.6.
We split the datasets randomly into three portions 'training', 'validation', and 'test' sets,

each containing 60%, 20%, and 20% respectively. We randomly partition the training set into the labeled and the unlabeled set with a ratio of 1:4. This is done to have the labeled and unlabeled data from a similar distribution. We further simulate the low-resource training by reducing the number of samples available in the labeled set. The amount of labeled data is varied between 5%, 10%, 20%, 30%, 50%, 75%, and 100% of the available labeled set. Table 4.1 provides the statistics of the number of samples in the labeled and the unlabeled sets.

	Labeled set							**Unlabeled set**
	100%	**75%**	**50%**	**30%**	**20%**	**10%**	**5%**	**100%**
Founta	10.3K	7.7K	5.2K	3.1K	2.1K	1.0K	0.5K	41.3K
Davidson	3.0k	2.2K	1.5K	0.9K	0.6K	0.3K	0.15K	12.0K

Table 4.1: Statistics of the number of samples in the Founta and the Davidson labeled and unlabeled sets in simulated low-resource setup.

4.3.2 Sentence embeddings

The Universal Sentence Encoder (USE) offers two different implementations of architectures. One is based on the transformer (Vaswani et al., 2017) which provides high accuracy at a large computational cost, and the other is based on Deep Averaging Network (DAN) (Iyyer et al., 2015) which has reduced computational cost with accuracy being the trade-off. We use the transformer-based model[1]. This model is trained on data from Wikipedia, web news, web question-answer pages, and discussion forums. The sentence embedding space has a dimension of 512.

4.3.3 Label propagation

The label propagation API provided by scikit-learn library[2] is used. Euclidean distance is used to compute the distance between two data points. Based on the validation set, we have chosen 80 nearest neighbors and a maximum of 4 iterations to perform label propagation.

4.3.4 Model setup

We use an MLP with two hidden layers to transform the pre-trained USE embeddings to task-specific representations. We also use the same MLP architecture to perform the hate speech classification. As shown in Figure 4.1, the MLP model has pre-trained representations (USE embeddings) as input. The hidden layers have ReLU activation (Agarap, 2018). The transformed task-specific representations are taken from the 1^{st} or 2^{nd} hidden layer, referred to as 'h1 representations' or 'h2 representations' in our experiments, respectively. We use Adam (Kingma and Ba, 2015) optimizer, early-stopping

[1] https://tfhub.dev/google/universal-sentence-encoder-large/4
[2] https://scikit-learn.org/stable/modules/label_propagation.html

based on the validation set, and a maximum of 10 epochs. Both the hidden layers have 50 units. Hence, the task-specific representations obtained from the hidden layers 'h1' and 'h2' have the dimension of 50. The model weights learned while training the system with only the labeled data are used to initialize the classifier before training with labeled and unlabeled data.

4.4 Results and discussion

In this section, we discuss the results on:

1. Semi-supervised learning using pre-trained representations (USE embeddings) and task-specific representations for binary and multi-class classification of hate speech.
2. Qualitative and quantitative analysis of representations used for label propagation algorithm.
3. Semi-supervised learning with unlabeled samples chosen from a distribution different from training datasets' distribution.

We report the mean and standard deviation of macro-averaged F1 evaluated over five separate runs on the test set. Each run uses an MLP classifier with a different random weight initialization. The 95% confidence interval on macro-averaged F1 obtained using the paired bootstrap test (Dror et al., 2018; Efron and Tibshirani, 1994) for the three-class classification is ± 1.6 and ± 3.0 for the Founta and the Davidson test sets, respectively. Similarly, the confidence interval on macro-averaged F1 obtained for the binary classification is ± 0.6 and ± 1.8 for the Founta and the Davidson test sets, respectively.

4.4.1 Semi-supervised learning using label propagation

In this sub-section, we present and discuss the results obtained for the semi-supervised learning using label propagation. We compare the pre-trained embeddings against the task-specific representations as data representations for the label propagation algorithm. The activation of hidden layers of the MLP classifier is used as task-specific representations for the label propagation algorithm.

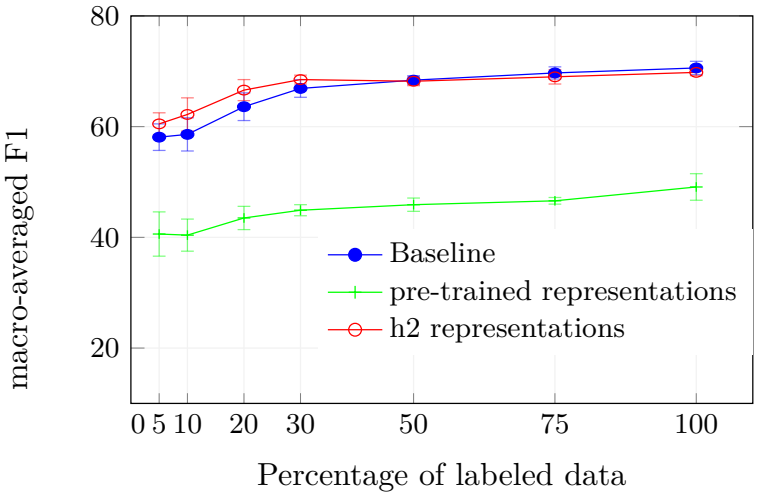
Figure 4.2 and Figure 4.3 show macro-average F1 obtained for the semi-supervised learning approach for multi-class classification and binary classification, respectively. The results are obtained on the Founta and the Davidson test sets. For the semi-supervised setup, the amount of unlabeled data is unchanged, but the amount of labeled data is varied to 5%, 10%, 20%, 30%, 50%, 75% and 100% of the available labeled set, thus simulating low-resource labeled data scenarios. The 'Baseline' results are obtained by training the MLP classifier with only the labeled set, and without using the unlabeled data and the label propagation. The 'pre-trained representations' and 'h2 representations' show the results of the MLP classifier trained using the combined labeled and unlabeled set, where, the labels for the unlabeled data were obtained using the label propagation algorithm. For the 'pre-trained representations', we used the pre-trained USE embeddings as data representations of the labeled and unlabeled sets for the label

propagation algorithm. Likewise, for 'h2 representations', we used the activations of the second hidden layer of the MLP model trained on labeled data as data representations for the label propagation algorithm. For our semi-supervised learning using task-specific representations, we obtained similar results for both 'h1 representations' and 'h2 representations', thus, further in this chapter, we only present the results obtained from the 'h2 representations'. In all four cases, to train the MLP classifier, the pre-trained USE embeddings were used as input to the model. The terminologies used in figures are summarized in Table 4.2.

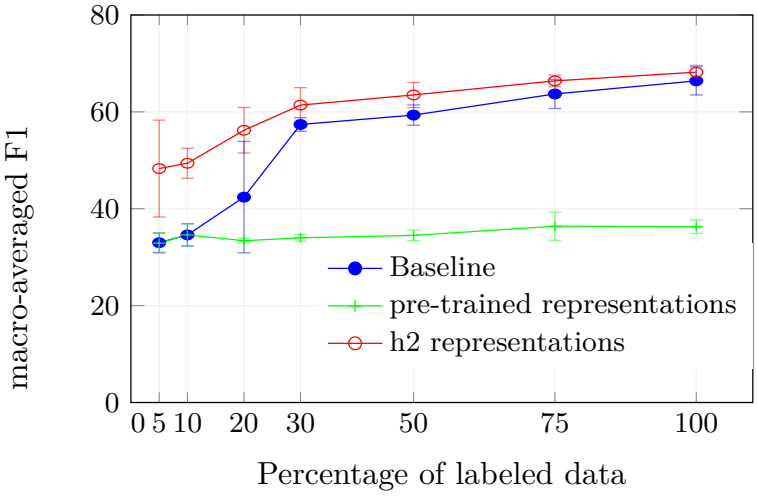
	Labeled data	**Unlabeled data**	**Representations for the label propagation**
Baseline	✓	✗	–
pre-trained representation	✓	✓	pre-trained USE embeddings
h1 representations	✓	✓	activation of the first hidden layer('h1') of the MLP
h2 representations	✓	✓	activation of the second hidden layer('h2') of the MLP

Table 4.2: Summary of terminologies used to present the results.

In Figure 4.2 and Figure 4.3, for both datasets, we observe that the 'Baseline' results improve with an increase in the number of labeled samples used for training the system. For the 'Baseline' results, for multi-class classification, training with 100% of the labeled data gave a macro-averaged F1 and standard deviation of 70.6±1.2 and 66.4±2.9 for the Founta and the Davidson datasets, respectively. Similarly, for the binary classification, we obtained the macro-averaged F1 and standard deviation of 91.3±0.1 and 88.3±0.3 for the Founta and the Davidson datasets, respectively. As expected, the performance of the binary classification task is significantly higher than the performance of the multi-task classification task for both datasets. This is due to the fact that the fine-grained classification of hate speech into 'hateful', 'abusive', and 'normal' classes is more challenging than the binary classification of 'toxic speech' versus 'non-toxic speech'.
Furthermore, we observe that semi-supervised training by combining the labeled and the unlabeled data using pre-trained representations for the label propagation performs worse than the 'Baseline'. This implies that label propagation algorithm using pre-trained representations is unreliable, thus introduces significant noise when labeling the unlabeled data.
In Figure 4.2 and Figure 4.3, we observe that using 'h2 representations' for the label propagation gives results similar or better than the 'Baseline'. This shows that by training the MLP classifier with a few amount of labeled data we can transform the generic representations to task-specific representations that model class label information, and are thus effective for labeling the unlabeled data using the label propagation algorithm. For multi-class classification, compared to the 'Baseline' results, using 'h2 representations', we observe a relative improvement up to 6.1% (58.6 to 68.2) using 10% of the



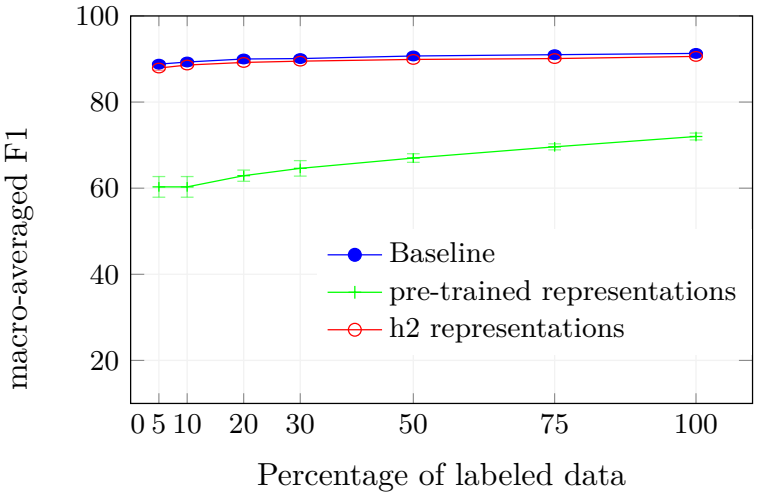
(a) Founta



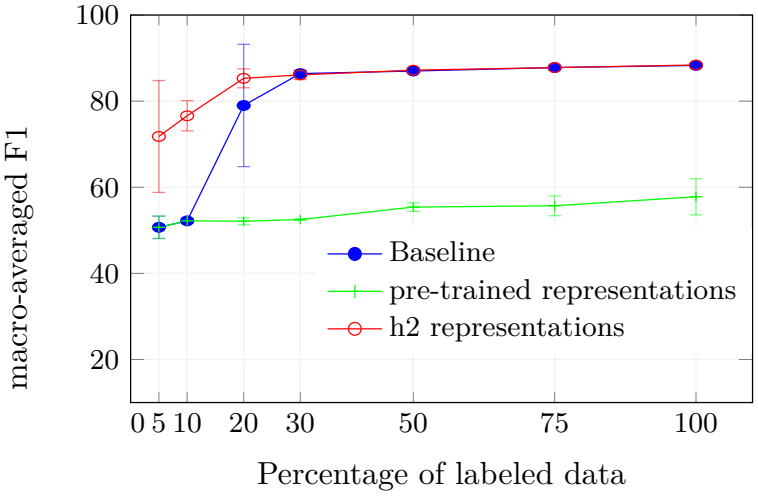
(b) Davidson

Figure 4.2: Semi-supervised learning using label propagation. Macro-average F1 on the test set obtained using MLP classifier for *multi-class* classification.

labeled data for the Founta dataset. Similarly, 46.4% (33.0 to 48.3) relative improvement using 5% of the labeled data of Davidson dataset. For both the datasets, we observe that the performance gain reduces as the amount of available labeled samples increases. The results in Figure 4.2 indicate that the label propagation-based semi-supervised learning is effective especially when the number of available labeled samples is small.
In Figure 4.3, for binary classification, we observe that semi-supervised training using 'h2 representations' gave similar results as 'Baseline' for the Founta dataset, for all the cases of low-resource scenarios. This shows that the binary classification task being an



(a) Founta



(b) Davidson

Figure 4.3: Semi-supervised learning using label propagation. Macro-average F1 on the test set obtained using MLP classifier for *binary* classification.

easier task can perform well with a fewer amount of labeled samples, and thus provides no improvements with the semi-supervised learning. However, compared to the 'Baseline' results of the Davidson dataset, we observe up to 41.6% (50.7 to 71.8) relative improvements when 5% of the labeled data is used for semi-supervised learning using 'h2 representations'. The smaller dataset size of the Davidson dataset can be attributed to the improvement in the binary classification performance in low resource scenarios.
Overall, our results show that semi-supervised learning using task-specific representations improves the final classification performance, especially in the low-resource scenarios.

These task-specific representations can be obtained by training an MLP classifier with a small amount of available labeled data.
We note that the label propagation algorithm works by constructing a $(l+u) \times (l+u)$ probabilistic transition matrix T, where l is the number of samples in the labeled set, and u is the number of samples in the unlabeled set. The size of this matrix T quadratically increases when the number of samples in the labeled or the unlabeled set increases. Thus, unlike neural networks, where the size of the model remains fixed, for the label propagation, the increase in the number of labeled or unlabeled set increases leads to quadratic increase in memory and computational overhead.

4.4.2 Analysis of pre-trained and task-specific representations

In this subsection, we analyze the difference between the pre-trained and the task-specific representations using intra-class and inter-class distances. We also qualitatively analyze these representations using color plots and Principal Component Analysis (PCA) (Karl Pearson F.R.S., 1901) plots.

	pre-trained representations (USE embeddings)	**h1 representations**	**h2 representations**
intra-class distance			
'normal'	1.33	0.89	1.48
'abusive'	1.26	0.81	1.43
'hate'	1.25	0.87	1.63
inter-class distance			
'normal'-'abusive'	1.33	1.17	2.72
'normal'-'hate'	1.33	1.14	2.39
'abusive'-'hate'	1.33	1.14	2.99

Table 4.3: Average inter-class and intra-class euclidean distance across different representations for the training set of Founta dataset.

The euclidean distance δ_{ij} between two samples x_i and x_j having Z features can be computed as below:

$$\delta_{ij} = \sqrt{\sum_{z=1}^{Z} (x_i^z - x_j^z)^2}$$

The average intra-class distance measures the average distance between every samples in the same class, for a given class label C containing n samples, average intra-class distance $\Delta(C)$ can be computed as:

$$\Delta(C) = \frac{1}{n \times (n-1)} \sum_{\substack{i,j \in C \\ i \neq j}} \delta_{ij}$$

The average inter-class distance measures the average distance between every samples in the class C_1 and the class C_2, for a given class label C_1 and C_2 containing m and n samples, respectively, average inter-class distance $\Delta(C_1, C_2)$ can be computed as:

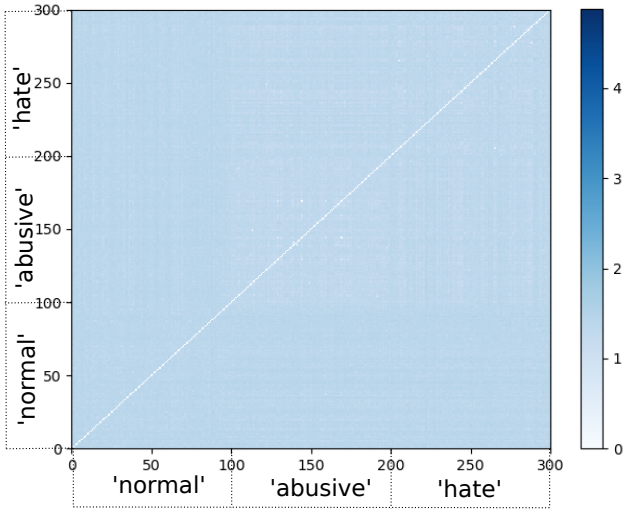
$$\Delta(C_1, C_2) = \frac{1}{m \times n} \sum_{i \in C_1, j \in C_2} \delta_{ij}$$

Table 4.3 shows the average intra-class and inter-class euclidean distance across different representations for the Founta training set, containing the labeled and unlabeled set. The 'h1 representations' and 'h2 representations' are obtained from the MLP model trained only on the labeled data. As we have split the training set into labeled and unlabeled sets, the ground truth labels are available. For our analysis, these ground truth labels are used as class labels of labeled and unlabeled sets.

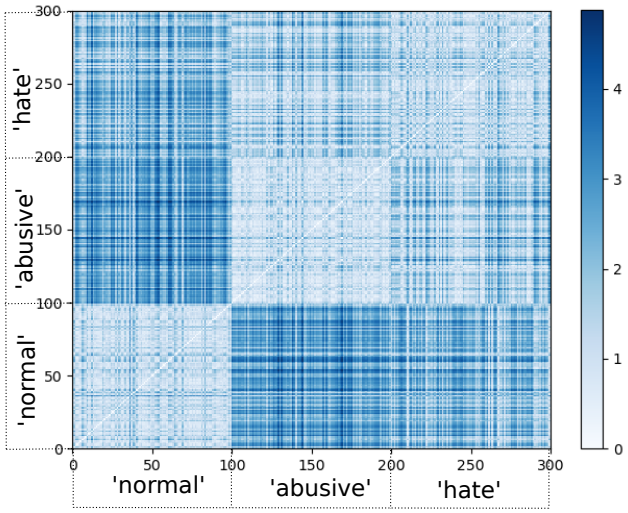
From Table 4.3, we observe that the intra-class and inter-class distances are similar for the 'pre-trained representations'. This implies that the pre-trained representations belonging to the samples from one class overlap with samples belonging to the other classes. This also shows that the pre-trained embeddings are generic and unaware of task-specific information. However, the task-specific representations ('h1 representations' and 'h2 representations') have lower intra-class and higher inter-class distances, this shows that the task-specific representations obtained by training the MLP model with a small amount of labeled data capture the task-specific information. Label propagation algorithm is efficient when the samples belonging to the same class are close to each other and far away from the samples belonging to the other classes. Thus, in Section 4.4.1, we observed that using task-specific representations help label propagation to efficiently label the unlabeled set.

Further, we qualitatively analyze the euclidean distances across 'pre-trained representations' and 'h2 representations' using color maps, shown in Figure 4.4a and Figure 4.4b, respectively, for the Founta labeled set. We randomly consider 300 samples from the Founta dataset, of which the 1^{st} set of 100 samples belongs to the labeled set of 'normal' class, the 2^{nd} and the 3^{rd} set of 100 samples belong to the labeled sets of 'abusive', and 'hate' classes respectively. From Figure 4.4a, we observe that the pre-trained representations has a similar distance across all the samples, thus appearing in a similar color. This infers that pre-trained representations are generic. However, as shown in Figure 4.4b, the intra-class distance of the samples within 'normal' and 'abusive' classes is smaller than their inter-class distance of 'h2 representations', hence appearing as lighter-colored squares. From these figures, we observe that 'h2 representations' capture task-specific information.

In Figure 4.5, we compare the PCA plot of the 'pre-trained representations' (USE embedding) of labeled and unlabeled data against the PCA plot of 'h2 representations' of labeled and unlabeled data obtained from the MLP model trained on labeled data. To visualize the pre-trained and the task-specific representations using a 2-dimensional plot, we perform a dimensionality reduction using the PCA. The labeled and unlabeled data are chosen from the Founta training dataset. To plot the data, we consider 50 random samples from each class for the labeled set and 50 random samples from each class for



(a) pre-trained representations (USE embeddings)

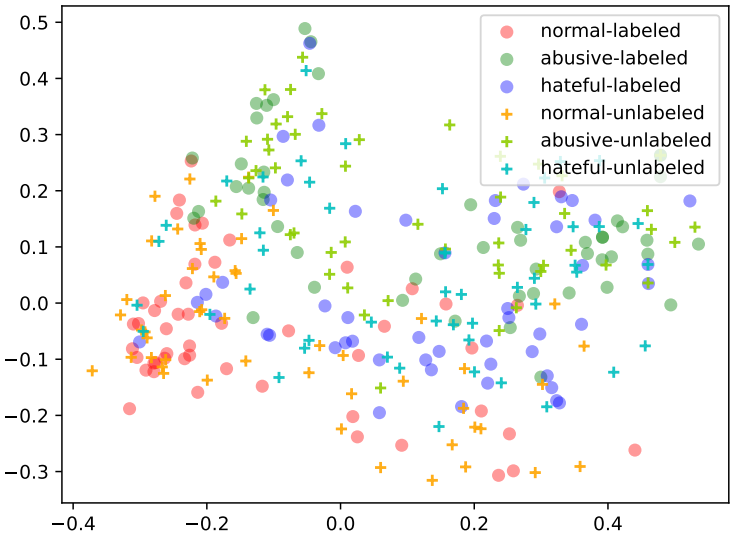


(b) h2 representations

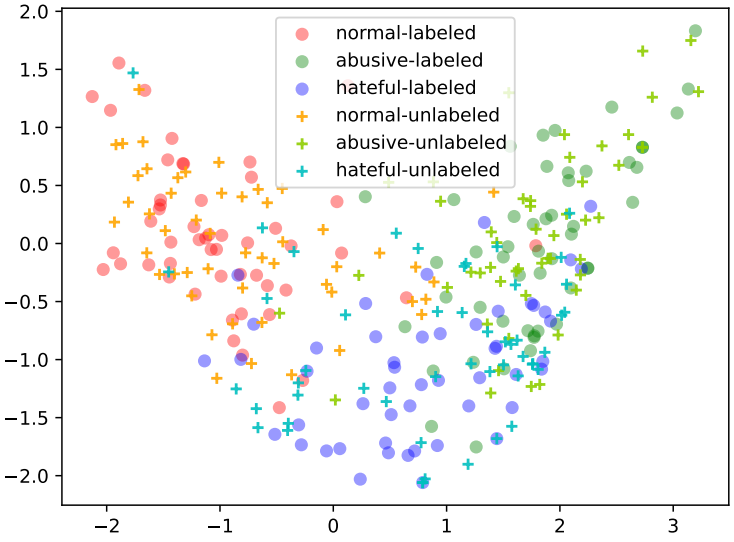
Figure 4.4: Color plots for distances between samples. Founta labeled set.

the unlabeled set. For the unlabeled set, ground truth labels are used to represent the samples in the plot.

In Figure 4.5a, for 'pre-trained representations', we observe that labeled and unlabeled samples from one class are not well separated from samples from other classes. Especially, we observe that the samples from the 'hateful' class overlap with the samples from other classes. However, in Figure 4.5b, for task-specific 'h2 representations', we observe that the samples belonging to the three classes are much more separated than the pre-trained USE embeddings. Thus, we can note that the pre-trained embeddings are generic, but, these pre-trained embeddings can be transformed to task-specific representations such



(a) pre-trained representations (USE embeddings)

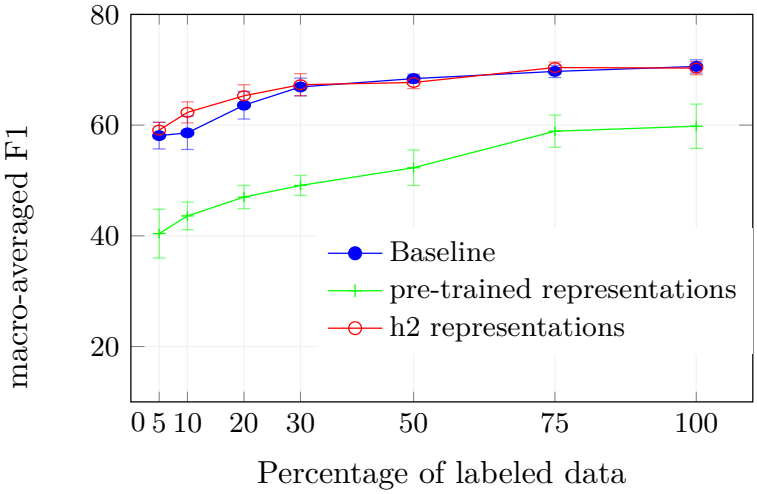


(b) h2 representations

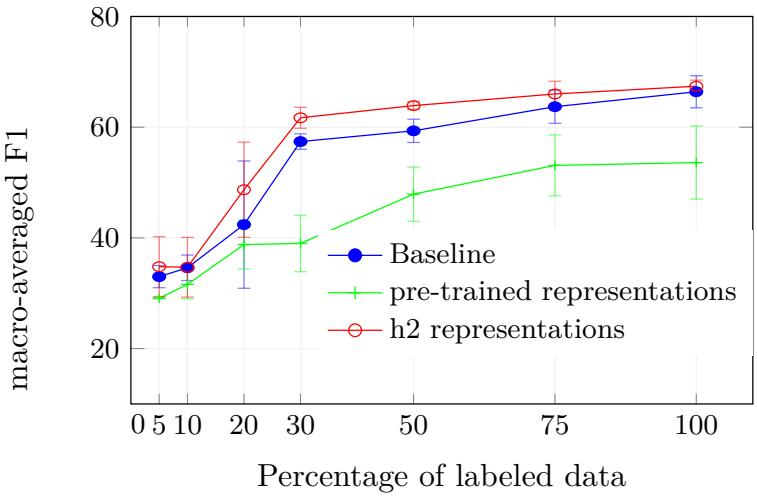
Figure 4.5: PCA plots of labeled and unlabeled data. Founta training set.

that data belonging to one class is more separated from other classes, and can thus be helpful for label propagation-based semi-supervised learning for hate speech classification.

4.4.3 Semi-supervised learning using unlabeled data from different distribution



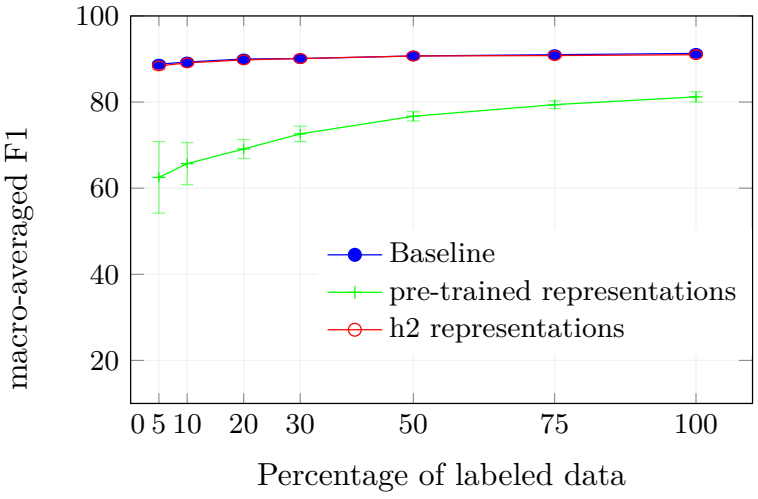
(a) Founta



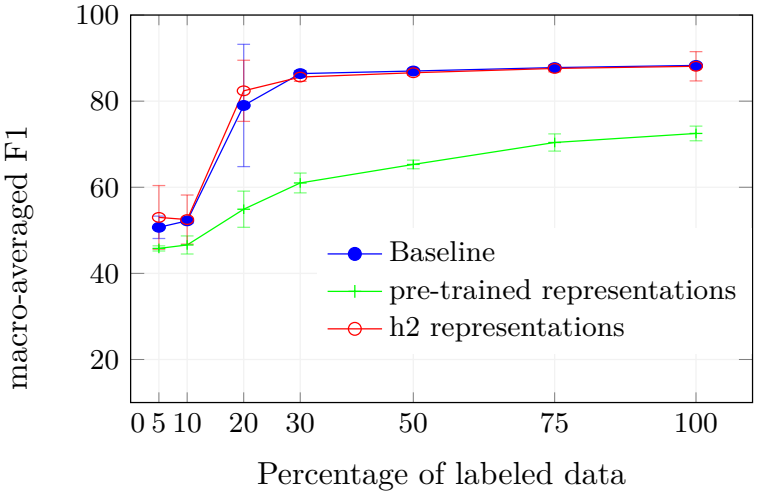
(b) Davidson

Figure 4.6: Semi-supervised learning using label propagation for *multi-class* classification. 'Unlabeled web crawl tweets' is used as unlabeled data. Macro-average F1 on the test set obtained using MLP classifier.

The results obtained in Section 4.4.1 evaluate the semi-supervised learning when the unlabeled samples belong to the same distribution as the labeled set. We achieved it by splitting the original training data into the labeled and the unlabeled sets. In this subsection, our unlabeled set is a set of unlabeled tweets which is not part of the Davidson or the Founta datasets. We use a part of the tweets sampled by Yang and Leskovec (2011),



(a) Founta



(b) Davidson

Figure 4.7: Semi-supervised learning using label propagation for *binary* classification.. 'Unlabeled web crawl tweets' is used as unlabeled data. Macro-average F1 on the test set obtained using MLP classifier.

a large collection of 580M tweets, collected over eight months from June 2009. Further in this section, we refer to these tweets as 'unlabeled web crawl tweets'. For our study, to compare results with results in Section 4.4.1, thus to have the same number of unlabeled samples, we randomly sample 41.3K and 12.0K tweets and use them as unlabeled sets for semi-supervised training on Founta and Davidson datasets, respectively.
Figure 4.6 and Figure 4.7 show macro-average F1 of the semi-supervised approach using 'unlabeled web crawl tweets' for multi-class and binary classification, respectively. The

results are obtained on the Founta and the Davidson test sets. Similar to the results in Section 4.4.1, we observe that the semi-supervised training using the pre-trained representations gave poorer results compared to the 'Baseline' results of training using only the labeled set for both multi-class and binary classification tasks. However, using 'h2 representations', we obtain similar or better results compared to 'Baseline' results.
For multi-class classification, compared to the 'Baseline' results, by using 'h2 representations', we observe relative improvements of up to 6.3% (58.6 to 62.3) using 10% of the labeled data and 14.8% (42.4 to 48.7) using 20% of the labeled data for the Founta and the Davidson datasets, respectively. In comparison, using the unlabeled data from similar distribution (in Section 4.4.1), we had obtained similar results, an improvements of 6.1% (58.6 to 68.2) for Founta test set using 10% of the labeled data, and relative improvements of 32.5% (42.4 to 56.2) using 20% of the labeled data for the Davidson dataset.
For binary classification task, we observe no improvements by using 'h2 representations' for the Founta dataset and a very small relative improvement of up to 4.3% (79.0 to 82.4) by using 20% of the labeled data for the Davidson dataset. In comparison, using the unlabeled data from similar distribution (in Section 4.4.1), we observe no improvements by using 'h2 representations' for the Founta dataset and a relative improvement of up to 6.7 % (79.0 to 85.3) by using 20% of the labeled data for the Davidson dataset. Compared to the results of using unlabeled data from a similar distribution as that of labeled data (results in Section 4.4.1), we obtain a lower performance improvements by using unlabeled data from a different distribution. However, very poor performance improvements in Davidson dataset compared to Founta dataset can be attributed to the fact that Davidson dataset is sampled using key-word based sampling, whereas, Founta dataset is sampled using random boosted-sampling, thus closer to the realistic distribution of the Tweets.
The results obtained in this subsection show that the label propagation-based semi-supervised learning using task-specific representations can be helpful to improve the hate speech classifier's performance even when the unlabeled data does not follow the distribution of the available labeled data. The results also indicate that semi-supervised learning can be helpful when we have very few labeled data. Our semi-supervised learning approach showed more improvements for multi-class classification than binary classification, and when the amount of available labeled data is small.

4.5 Conclusion

In this chapter, we have explored label propagation-based semi-supervised learning for hate speech classification. Using semi-supervised learning, we attempted to use unlabeled data along with a few labeled samples to improve the classification performance. We showed that label propagation using the pre-trained sentence embeddings yields poorer classification performance than training the classifier with only the labeled data. This is because pre-trained embeddings are generic representations. We used a multi-layer perceptron (MLP) to transform these pre-trained representations to task-specific representations using a small number of labeled samples. Our results showed that using

task-specific representations helped semi-supervised learning, especially in low-resource scenarios.
Furthermore, using intra-class and inter-class distances, we showed that the pre-trained representations (USE embeddings) have similar intra-class and inter-class distances. Hence, the samples are not well separated in the pre-trained embedding space. However, the task-specific representations had lower intra-class distance and higher inter-class distance, which makes the label propagation algorithm transduce reliable labels from the labeled data to unlabeled data as the labeled and unlabeled samples from the same class are close to each other in the task-specific representations space.
Finally, we used the unlabeled samples from Twitter which had a different distribution than the considered datasets for our label propagation-based semi-supervised training. The experiments revealed improvements in classification performance for multi-class classification in low-resource scenarios.
We conclude that semi-supervised learning based on label propagation using task-specific representations helps to improve the multi-class classification of hate speech in low resource scenarios.

5 Data augmentation for hate speech classification

5.1 Introduction

In this chapter, the conditional language model-based data augmentation for hate speech classification is introduced. The performance of the deep neural network-based classifier depends on the amount of available labeled data to train the system. However, to train a reliable classifier for the hate speech classification task, most of the available hate speech corpora have a very low amount of labeled data. This is because of the expenses involved in collecting and manually annotating the data. Thus, data augmentation techniques can be employed to increase the amount of training data by creating synthetic samples. Data augmentation techniques expand the training set by creating new synthetic samples using the existing data. These synthetic samples add new information to the training data to boost the model performance (Shorten and Khoshgoftaar, 2019).

5.1.1 Data augmentation in natural language processing

Data augmentation techniques are largely used in computer vision and speech applications. In computer vision, it involves simple transforms like scaling, rotating (Simard et al., 1998), cropping (Howard, 2013), etc. Likewise, in speech, it involves transforms like warping the signals (Jaitly and Hinton, 2013; Ko et al., 2015), etc. Unlike computer vision or speech, in Natural Language Processing (NLP) such simple transforms do not retain the syntax or the semantics of a given sentence, hence, cannot be easily applied. For example, given an image of a cat, applying transforms such as rotating or flipping still retains an image of the cat. However, given the sentence "the earth is greater than the moon" if we apply the random crop, the sentence may transform to "is greater than the moon", losing its original meaning. Similarly, transforms such as shuffling the sentence may transform the sentence to "the moon is greater than the earth" keeps the grammatical correctness of the sentence, but changes the original meaning of the sentence, and thus may not retain the corresponding label. Likewise, similar to flipping an image, if we flip the given sentence, it turns into "moon the than greater is earth the", thus losing the grammatical correctness and the meaning. Hence, alternative techniques for data augmentation are explored in the field of NLP.
In NLP, various data augmentation approaches are explored. One group of approaches include replication of samples by performing minor perturbations in the text, such as the addition of a random word, deletion of a word, swapping of words in a sentence, and replacing words with synonyms (Wei and Zou, 2019). Replacing the words with

synonyms based on an external knowledge-base such as WordNet (Miller, 1995) was explored by Jungiewicz and Smywiński-Pohl (2019); Zhang et al. (2015); Abdollahi et al. (2020). Some approaches in this group replicate samples by replacing words based on semantic similarity using the pre-trained embedding representations (Wang and Yang, 2015; Abdollahi et al., 2020), such as Word2Vec (Mikolov et al., 2013a,b) embeddings or Global Vectors for word representation (GloVe) (Pennington et al., 2014) embeddings. Some authors went beyond and chose to replace words based on the surrounding context by using contextual embedding models (Kumar et al., 2020) such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019). For classification tasks, Kobayashi (2018); Wu et al. (2019) used class label information along with the embeddings to replace words by ensuring class label compatibility.
Another group of approaches has explored translation-based methods such as back-translation (Sennrich et al., 2016; Shleifer, 2019). Back-translation involves translating a given text from a language A to a different language B and then translating back from language B to language A.
Last group of approaches involves generating the synthetic text using auto-encoders (Hu et al., 2017; Anaby-Tavor et al., 2020; Kumar et al., 2020) based models such as Variational Auto-Encoder (VAE) (Kingma and Welling, 2014) or BERT, sequence-to-sequence (seq2seq) model such as Bidirectional and Auto-Regressive Transformers (BART) (Lewis et al., 2020), and auto-regressive language models such as Generative Pre-Trained Transformer-2 (GPT-2)(Radford et al., 2019).

5.1.2 Data augmentation for hate speech classification

Similar data augmentation techniques have been explored in the domain of hate speech classification to increase the number of training samples. Few authors have demonstrated the effectiveness of data augmentation using text perturbation based methods. Simple perturbation on the text like inserting a random word, deleting a random word, or replacing based on synonyms from WordNet was performed by Luu et al. (2020) for data augmentation, and improvements was shown on Vietnamese hate speech classification datasets. Word replacement based on features from the external knowledge base such as ConceptNet (Liu and Singh, 2004) and Wikidata knowledge graphs were explored by Sharifirad et al. (2018). A knowledge graph is a structured form of data showing a word and its associated real-world concepts. For example, for a given word "**computer**", Wikipedia knowledge graph shows related concepts as "*instance of* **tool**", "*uses* **electricity**", etc. Going beyond simple synonym based word replacement, Rizos et al. (2019); Benk (2019) demonstrated the use of pre-trained embeddings and similarity-based methods to replace the words in a given sentence to replicate the sentences.
Data augmentation based on text transformation using back-translation was explored by Aroyehun and Gelbukh (2018) to improve hate speech classification. Beddiar et al. (2021) performed paraphrasing on the back-translated sentences to obtain additional synthetic samples, and obtained higher performance than using only back-translation.
Approaches based on synthetic sample generation of hate speech text using language modeling approach by using Long Short-Term Memory (LSTM) (Hochreiter and Schmid-

huber, 1997) and GPT-2 (Radford et al., 2019) are explored by Rizos et al. (2019); Wullach et al. (2020). Rizos et al. (2019) compared sample generation using LSTM based language model with word replacement based on embeddings for data augmentation, and showed that sample generation using LSTM model for augmentation did not show any performance improvements, while word replacement based augmentation gave an improved results compared to not performing augmentation. However, compared to the LSTM based sample generation approach of Rizos et al. (2019), Wullach et al. (2020) used the GPT-2 model for sample generation thus taking the advantage of the pre-trained language model and showed improvements in the classification results on several datasets. Generating the data using a language model involves training a language model with the training data belonging to a specific class, thus requiring multiple models to generate data corresponding to different class labels. Thus, to use a single GPT-2 model for sample generation, class conditioning during decoding by modifying the hidden states of the model is explored by Liu et al. (2020). The authors have shown the effectiveness of this method in low-resource setting. Adversarial methods of jointly training a language model along with discriminator to classify hate speech was explored by Cao and Lee (2020), where Sequence generative adversarial networks (SeqGAN) (Yu et al., 2017) based generator and the deep neural network-based discriminator was jointly trained using reinforcement learning. Although, data augmentation using adversarial method showed improvements, the performance improvements achieved by Wullach et al. (2020) using GPT-2 model for sample generation was slightly higher.

5.1.3 Our approach for data augmentation

Given the significant improvements in the classification performance using the language generation-based data augmentation methods, we follow the pre-trained model language generation-based data augmentation approach of Wullach et al. (2020). This method involves fine-tuning two GPT-2 models one for the 'hate speech' class and the other for the 'non-hate speech' class. They generate 600K samples for each class. Furthermore, they filter the samples using a BERT model fine-tuned on the training data to keep only the top 100K samples. The filtered samples are then used to augment the original training set and train a classification model for the binary classification task. Compared to the approach of Wullach et al. (2020), our approach (D'Sa et al., 2021) for data augmentation using language generation has two key differences:

1. Instead of simple binary classification of 'hate speech' versus 'non-hate speech', we attempt a three-class classification of 'hate speech', 'abusive language', and 'normal speech', which is known to be more complex task due to overlap between hate speech and abusive language (Davidson et al., 2017; D'Sa et al., 2020).
2. Inspired by Keskar et al. (2019), we fine-tune a single class conditioned GPT-2 language model, as opposed to class-specific fine-tuning of multiple GPT-2 models of Wullach et al. (2020). By using a single class-conditioned language model, we reduce the number of models used to generate the data.

Our approach differs from the method of Liu et al. (2020). In our approach, we fine-tune a single GPT-2 model using conditional context token in the input text, however,

Liu et al. (2020) do not fine-tune the model, instead, they use a classifier to inject the conditional context at the decoding step of sample generation by the GPT-2 model, this requires the hidden state outputs of the model to be updated for every decoded token. Furthermore, we extend our work by exploring the effect of the quantity and the quality of the generated data required to improve the classification performance. We study the capabilities of our data augmentation technique in a low resource setting by training our model with a only few amount of available training data. We also empirically evaluate the influence of using the BERT model for filtering the generated samples. We have performed our experiments on two widely used hate speech corpora- the Founta dataset (Founta et al., 2018) and the Davidson dataset (Davidson et al., 2017).
To summarize, the contributions of this chapter are:

1. We generate synthetic training samples using the objective of conditional language modeling for data augmentation for multi-class hate speech classification.
2. We analyze how the classification performance varies depending on the quantity of the augmented samples.
3. We study how the filtering of generated samples affects the final classification performance.
4. By simulating the low resource setting, we study the efficacy of our data augmentation technique when the amount of available training data is less.

The following contents of this chapter are organized as follows, Section 5.2 describes the objective of the conditional language modeling and our approach for data augmentation using conditional language modeling. Section 5.3 describes the datasets, the model, and the model parameters chosen for our experiments. We present and discuss our results in Section 5.4 and conclude in Section 5.5.

5.2 Data augmentation using language modeling

In this section, we describe the approach for data augmentation using the GPT-2 model. We use the objective of conditional language modeling to train the GPT-2 model and generate new synthetic samples for data augmentation.

5.2.1 Conditional language modeling

A typical language modeling task involves learning the joint probability distribution of a sequence (Bengio et al., 2003). Given the vocabulary V containing a fixed set of distinct tokens and a sequence $z^j = (z_1^j, z_2^j, ..., z_n^j)$ of length n, where $z_i^j \in V$, and D containing m samples $D = \{z^1, z^2, ..., z^m\}$ the joint probability distribution of the sequence is given as:

$$p(z^j) = \prod_{i=1}^{n} p(z_i^j | z_1^j, \ldots, z_{i-1}^j) \tag{1}$$

A neural language model learns the parameter set θ such that it reduces the negative log-likelihood loss L over dataset D as:

$$L(D) = -\sum_{j=1}^{|D|} \log p(z^j|\theta) \tag{2}$$

Given a sub-sequence of t tokens $z = (z_1, z_2, \ldots, z_t)$, the learned parameter set θ can be used to sequentially generate next l tokens for a new sequence $\hat{z}$ as:

$$\begin{aligned}
\hat{z}_{t+1} &\sim p(z_{t+1}|z_1, \ldots, z_t; \theta), \\
\hat{z}_{t+2} &\sim p(z_{t+2}|z_1, \ldots, z_t, \hat{z}_{t+1}; \theta), \\
&\ldots \\
\hat{z}_{t+l} &\sim p(z_{t+l}|z_1, \ldots, z_t, \hat{z}_{t+1}, \ldots, \hat{z}_{t+l-1}; \theta)
\end{aligned}$$

The language model can be trained with a conditional context, given a token c as conditional context, equation (1) can be extend to:

$$p(z^j|c) = \prod_{i=1}^{n} p(z_i^j|c, z_1^j, \ldots, z_{i-1}^j)$$

Likewise, equation (2) extends to:

$$L(D) = -\sum_{j=1}^{|D|} \log p(z^j|c; \theta)$$

Given a sub-sequence of t tokens $z = (z_1, z_2, \ldots, z_t)$ and the conditional context token c, the learned parameter set θ can be used to sequentially generate next l tokens for a new sequence $\hat{z}$ as:

$$\begin{aligned}
\hat{z}_{t+1} &\sim p(z_{t+1}|c, z_1, \ldots, z_t; \theta), \\
\hat{z}_{t+2} &\sim p(z_{t+2}|c, z_1, \ldots, z_t, \hat{z}_{t+1}; \theta), \\
&\ldots \\
\hat{z}_{t+l} &\sim p(z_{t+l}|c, z_1, \ldots, z_t, \hat{z}_{t+1}, \ldots, \hat{z}_{t+l-1}; \theta)
\end{aligned}$$

In our study, we prepend every sample with its class label as the conditional context for the sentence.

5.2.2 Proposed methodology

Figure 5.1 shows the block diagram of our approach. We fine-tune a single pre-trained GPT-2 model using the conditional language modeling objective. To generate a large amount of diverse data that represents the distribution of the training set, we then use the fine-tuned GPT-2 model. We then filter the generated samples using a Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) model which

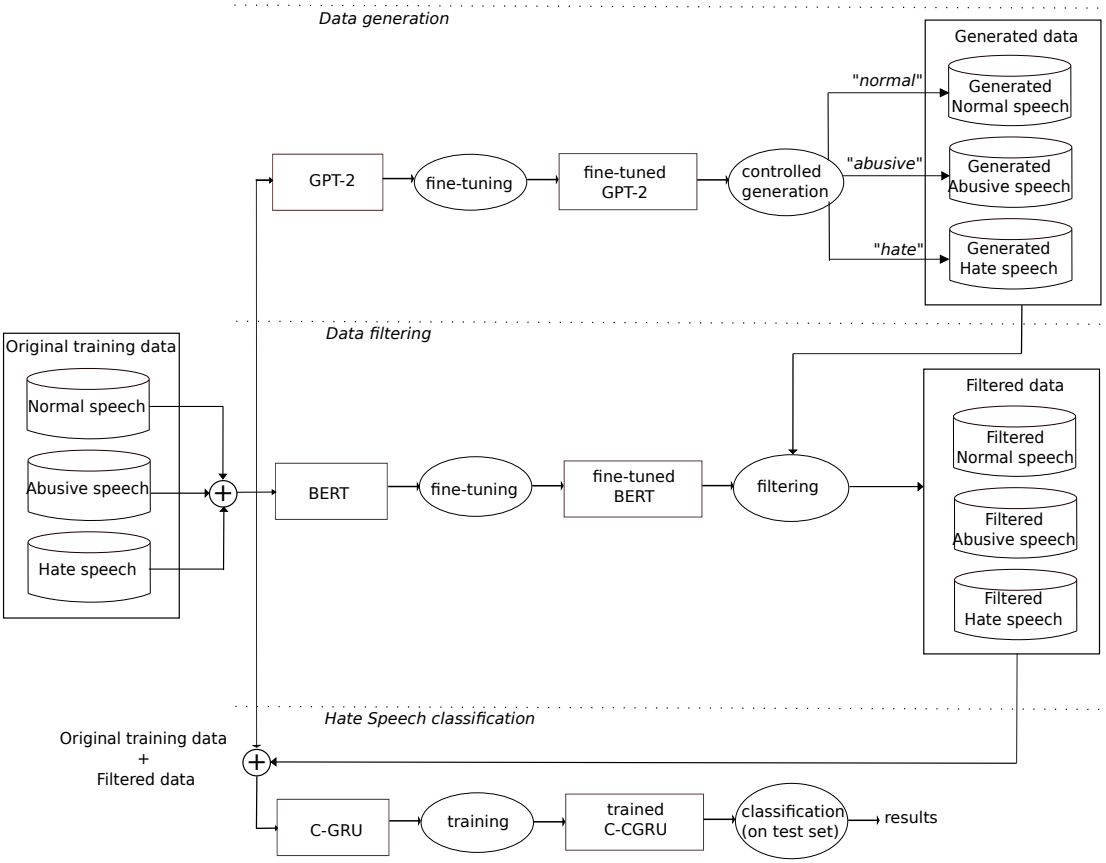


Figure 5.1: Block diagram for training an improved classifier with data augmentation.

has been fine-tuned on the original training set. The top-N generated samples sorted by the BERT model are added to the original training set. After this, a Convolutional-Gated Recurrent Unit (C-GRU) (Zhang et al., 2018) based classifier (explained in Section 3.2.5) is trained for multi-class classification of hate speech. Further in this section, we detail the steps involved in fine-tuning the GPT-2 model, generating synthetic data using GPT-2 model, filtering the generated data using the BERT model, and multi-class classification using the C-GRU model.

5.2.2.1 GPT-2 fine-tuning and data generation

To generate a large number of synthetic samples for data augmentation, we use an auto-regressive pre-trained GPT-2 model. To generate the samples that are similar to our training set, we fine-tune the GPT-2 model on the original training set. However, to fine-tune the single GPT-2 model for a given dataset, and to generate the data for each class separately, we train the GPT-2 model by conditioning it on the class labels. We achieve this by prepending the class label of the sample as a conditional context. For example, given a 'normal' class sentence such as *"a cat is sitting on the mat"*, we transform

it to *"**normal** a cat is sitting on the mat"* before using it as an input to fine-tune GPT-2 model.
We then use the fine-tuned GPT-2 model to generate a large number of new synthetic samples for every class. As we have used the class label as the conditional context to fine-tune the GPT-2 model, we use the class label as the prompt text to generate the data for the required class.

5.2.2.2 Filtering the generated sequences

Sometimes, the generated data can belong to a different class than expected, these samples when augmented to the training set can add noise to the final classifier. Thus, we use a technique adopted by Wullach et al. (2020) to filter the generated samples. Filtering involves ranking the generated samples and retaining only top rank samples. This technique removes the low-rank data than may not belong to the desired class. For our approach, filtering is performed by fine-tuning the BERT model. Most of the hate speech datasets have an imbalanced class distribution. For our considered datasets, the 'hate speech' class is minority class. The majority class samples can induce bias to the BERT classifier, where the model tends to favor a given sample to be predicted as the majority class. Thus, to avoid the bias induced by the majority class samples, to have an equal number of training samples for all the classes, we downsample the majority classes to have an equal number of samples as that of the minority class. We then fine-tune the BERT model for multi-class classification task with the downsampled dataset. The samples generated by the fine-tuned GPT-2 model are then used as input to the fine-tuned BERT model to sort them according to the softmax prediction score. We then retain only the top-N samples scored by the BERT model for a given class for the data augmentation.

5.2.2.3 Hate speech classifier

As presented by Zhang et al. (2018), the Convolutional-Gated Recurrent Unit (C-GRU) based architecture is a powerful hate speech classifier. This model is faster to train since it has fewer model parameters in comparison to the transformer-based BERT model, and is efficient for the hate speech classification. Furthermore, models such as BERT contains diverse information and representations captured from the large amount of web data during the pre-training stage. This prevents from a fair analysis of performance gains obtained due to the augmented data. Thus, as adopted by Wullach et al. (2020), we adopt a similar architecture for our hate speech classification. In the C-GRU based architecture, the input is first passed through convolutional layers, then followed by the Gated Recurrent Unit (GRU) layer.

5.3 Experimental setup

In this section, we describe the datasets considered for our data augmentation approach and the choice of hyper-parameters for the considered models.

5.3.1 Data description

For the multi-class classification of hate speech, we chose two datasets containing tweets sampled from Twitter, one by Founta et al. (2018) and the other by Davidson et al. (2017), here onwards, referred to as 'Founta' and 'Davidson'. A detailed description of the considered datasets is provided in Section 3.5. The 'hateful' tweet samples are in the minority in both datasets. For both these datasets, we have applied the text processing described in Section 3.6.
Each dataset is randomly split into three sets, 'training', 'validation', and 'test', containing 60%, 20%, and 20% of samples respectively. The training data is used to fine-tune the GPT-2 model for the sample generation, it is also used to fine-tune the BERT model for filtering the generated samples. The original training data is augmented with the filtered generated samples to train the C-GRU for the hate speech classification task. We use the validation set to tune the model parameters and perform early stopping. The test set is used to report the final performance of the C-GRU classifier.

5.3.2 Model parameters

To train the GPT-2, BERT, and C-GRU models, we use the model parameters used in the works of Wullach et al. (2020). We use the implementation of huggingface's transformers API (Wolf et al., 2020) to fine-tune the 'GPT-2 large' model.[1] The GPT-2 large has 24 transformer decoder layers containing 774M model parameters. To provide the conditional context, we prepend the input sentences with their class labels. We use a learning rate of 5×10^{-5} to train our GPT-2 model. We fine-tuned two GPT-2 models trained with the objective of conditional language modeling, one on the Founta training set and the other on the Davidson training set. The final generative model is chosen based on the lowest loss computed on the validation set after each epoch. To generate the new synthetic samples, we use the fine-tuned GPT-2 model. The class label is used as a prompt text to the fine-tuned GPT-2 model to generate samples for the respective class. The average number of words in the Founta and the Davidson datasets is 16.9 and 14.1, respectively. Thus, we generate the sequence until the occurrence of the '<EOS>' token, a special token to indicate the end of the sequence or a maximum of 30 tokens to have sufficiently long sequence that represents the training set distribution. We obtained a similar results for our data augmentation experiments by generating a maximum of 50 tokens. While decoding the sequential text output, we use nucleus sampling (Holtzman et al., 2019) with top-p and temperature set to 0.9. Nucleus sampling selects top-N words such that their cumulative probability mass exceeds the threshold indicated by the value of top-p. This sampling strategy improves diversity and reduces less reliable distribution compared to top-k sampling or beam search sampling. Overall, we generate 600K samples for each class label, thus, a total of 1800K samples for each dataset.
To fine-tune the BERT model for the filtering, we used the pre-trained 'BERT-base-uncased' model trained on the English corpora. We fine-tuned two BERT models, one on the training set of Founta, another on the training set of Davidson. At the end of

[1] https://huggingface.co/gpt2-large

each epoch, the model is evaluated for macro-averaged F1 measure using the validation set. The best model is then used for filtering the data. We use a learning rate of 1×10^{-5} and Adam (Kingma and Ba, 2015) optimizer. The generated data is sorted according to the softmax score obtained by the fine-tuned BERT model.
For the C-GRU classifier, words occurring less than three times are considered as out-of-vocabulary words, and are replaced with an '⟨UNK⟩' token. The model learns embeddings of dimension 300 for the words in the vocabulary. It is then followed by a 1D-convolutional layer with 100 filters with the filter and stride size of 4. The output of the convolutional layer is max-pooled and then passed to a Gated Recurrent Unit (GRU) layer with 100 hidden units, followed by another max-pool layer. Finally, a dense layer with an output dimension of 3 is used for three-class classification. The ReLU (Agarap, 2018) activation is used after the pooling layers and softmax activation is used for the output layer. The model is trained for maximum of 10 epochs, using an initial learning rate of 1×10^{-3} and the Adam optimizer. For the C-GRU model, at the end of each epoch, the macro-averaged F1-measure is evaluated on the validation set to perform early stopping and choose the best model for classification on the test set.

5.4 Results and discussion

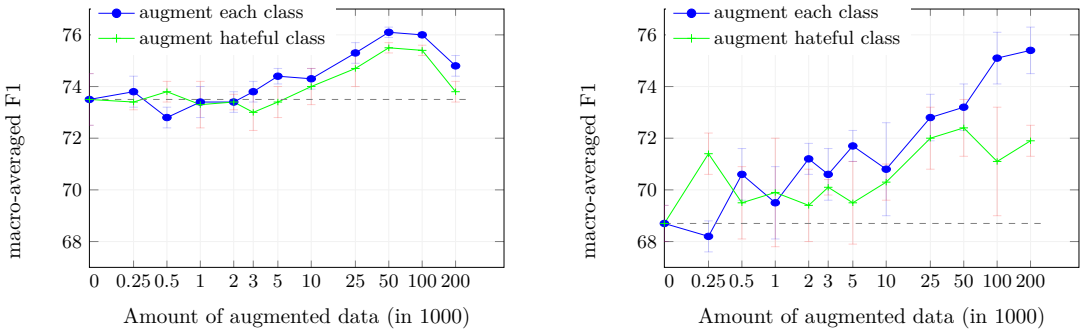
In this section, we discuss the results on:

1. Evaluation of the influence of the quantity of the data used for the data augmentation.
2. Evaluation of the quality of the generated data used for augmentation evaluated using classification technique.
3. Study of the influence of using the BERT model for filtering the generated samples.
4. Evaluation of the performance of data augmentation in low-resource training data scenarios.

We report the mean and standard deviation of macro-averaged F1 evaluated over five separate runs on the test set. Each run uses a C-GRU classifier with a different random weight initialization. The 95% confidence interval on macro-averaged F1 obtained using the paired bootstrap test (Dror et al., 2018; Efron and Tibshirani, 1994) is ±1.6 and ±2.8 for the Founta and the Davidson test sets, respectively.

5.4.1 Influence of quantity of data for augmentation

In this subsection, we analyze the influence of the quantity of the generated data used for augmentation to improve the performance of the hate speech classifier. We train the C-GRU classifier with the original training data combined with the varying amount of the synthetically generated data. We choose the top-N generated samples scored by the BERT model for the augmentation in the steps of 0.25K, 0.5K, 1K, 2K, 3K, 5K, 19K, 25K, 50K, 100K, and 200K samples. We have explored two strategies, (a) augmenting each class with an equal amount of generated data for each class, referred to as 'augment



(a) Founta (b) Davidson

Figure 5.2: Macro-averaged F1 on the test set. The C-GRU classifier is trained using the training data and different amount of augmented data (X-axis).

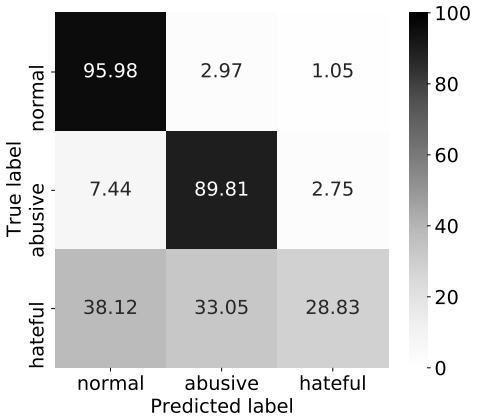
each class' in Figure 5.2 ; (b) augmenting data only for the 'hateful' class, referred to as 'augment hateful class' in Figure 5.2.

Figure 5.2a and Figure 5.2b show the macro-average F1 obtained on test sets by varying the amount of augmented data for the Founta and the Davidson datasets, respectively. The baseline macro-averaged F1 and standard deviation obtained using the C-GRU classifier trained without data augmentation is 73.5±1.0 for the Founta dataset and 68.7±0.7 for the Davidson dataset and correspond to the point 0 in X-axis of the Figure 5.2.

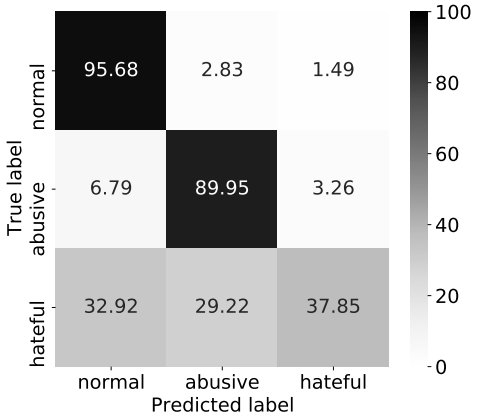
Figure 5.2a and Figure 5.2b show that data augmentation improves the classifier performance for the 'augment each class' and gives a relative improvement of up to 3.5% (73.5 to 76.1) for the Founta test set and up to 9.8% (68.7 to 75.4) for the Davidson test set. We observe performance gains even with few thousand samples augmented with the original training set. This shows that we can obtain classification improvement by increasing the number of training samples using the language model-based data augmentation.

In the 'augment hateful class' case, we observe a relative improvement to the classification performance by up to 2.7% (73.5 to 75.5) for the Founta test set and up to 5.4% (68.7 to 72.4) for the Davidson test set. We also observe improvement similar to 'augment each class' for the Founta test set and slightly lower performance than 'augment each class' for the Davidson test set. This shows that adding 'hateful' samples to the dataset improves the classification performance.

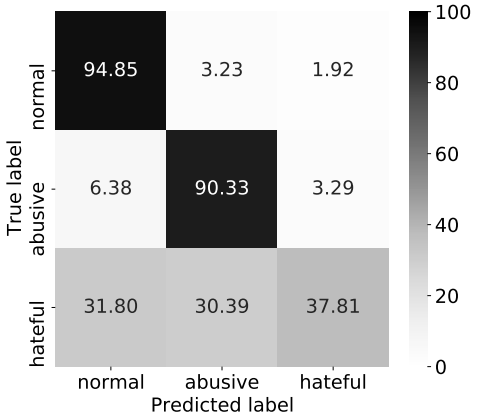
Figure 5.3a, Figure 5.3b, and Figure 5.3c show the confusion matrix obtained on the Founta test set for the C-GRU classifier trained on the original training set, for 'augment each class', and 'augment hateful class', respectively. To compare the confusion matrices, we augment 50K samples for each class for 'augment each class' and 50K samples for the 'hateful' class for 'augment hateful class'. Compared to the baseline confusion matrix, we observe an improvement for the true positives of the 'hateful' class for both 'augment each class' and 'augment hateful class'. This shows a significant improvement in classification performance of the 'hateful' class compared to 'normal' and 'abusive' classes. However, compared to the baseline confusion matrix, we observe a small decline in the



(a) 'no augmentation' (baseline)



(b) 'augment each class'



(c) 'augment hate class'

Figure 5.3: Confusion matrices on the Founta test set. Comparison of baseline results with 'augment each class' and 'augment hate class'.

true positive of the 'normal' class for 'augment hateful class', an increase in 'normal' and 'abusive' class samples being incorrectly predicted as 'hateful' class. This shows that adding samples only to the 'hateful' class biases the C-GRU model to predict the test set samples as 'hateful'.

Table 5.1, compares our approach using a single class conditional GPT-2 model with the approach of Wullach et al. (2020) implemented by us. For the approach of Wullach et al. (2020), we fine-tune three GPT-2 models, one for each class. We then generate the data for a given class from the model fine-tuned using the training data belonging to the respective class. We then filter the generated data using the BERT model and classify using the C-GRU classifier. We compare the results of augmenting 5K, 50K, and 100K

Amount of top-N generated data used for each class	Founta		Davidson	
	One GPT-2 model (our approach)	**Three GPT-2 models**	**One GPT-2 model (our approach)**	**Three GPT-2 models**
Baseline (no data augmentation)	73.5 ± 1.0		68.7 ± 0.7	
5K	74.4 ± 0.3	74.1 ± 0.4	71.7 ± 0.6	70.1 ± 0.6
50K	**76.1 ± 0.2**	75.1 ± 0.5	73.2 ± 0.9	73.6 ± 0.9
100K	76.0 ± 0.1	74.9 ± 0.4	**75.1 ± 1.0**	74.7 ± 0.4

Table 5.1: Macro-averaged F1 (± standard deviation) results on Founta and Davidson test sets. Comparison of our approach of fine-tuning a single GPT-2 model using the objective of conditional language modeling versus the approach of fine-tuning three GPT-2 models (one for each class) for the data generation.

samples to every class with the results obtained by our approach. From Table 5.1, we observe that our approach of using a single GPT-2 model fine-tuned with the objective of the conditional language modeling gave similar or slightly better results than approach of Wullach et al. (2020) for both the Founta and the Davidson datasets. This improvement in the classification performance can be attributed to fine-tuning a single GPT-2 model using more samples by combining the original training samples of all the classes than fine-tuning three GPT-2 models with less class-specific data. Thus, we have achieved a comparable performance by using three times less parameters (774M versus 2322M) to generate the synthetic training data by using a single class conditioned GPT-2 model.

5.4.2 Quality of augmented data

In this subsection, we analyze the quality of the generated synthetic samples from the conditional GPT-2 model used for the data augmentation. We fine-tune the GPT-2 model with the training set and generate a huge quantity of samples (600K samples for each class) from the fine-tuned GPT-2 model, we then filter the generated data using the BERT model fine-tuned on the training set. For the experiments in this subsection, we train a C-GRU classifier using only the top-N generated data filtered by the BERT model, without considering the original training data. To train the C-GRU classifier, we use an equal number of generated samples for every class. We evaluate our results on the Founta and the Davidson test sets.
Table 5.2 presents the classification performance of the C-GRU trained with only the generated and filtered data. The baseline results indicated in Table 5.2 are results obtained on the Founta and the Davidson test sets for the C-GRU model trained on the original training set. The results that are better than the baseline results are indicated in bold font. For both datasets, we observe an increase in the classifier's performance as the amount of generated data used to train the classifier increases. For the Founta dataset, we observe that the performance of the classifier trained with only the generated data is

Amount of top-N generated data used for each class	Founta test set	Davidson test set
Baseline (C-GRU trained on the original training data)	73.5 ± 1.0	68.7 ± 0.7
5K	69.2 ± 0.1	66.6 ± 0.6
10K	70.2 ± 0.8	68.7 ± 1.1
25K	71.4 ± 0.4	**69.9 ± 1.0**
50K	72.2 ± 1.1	**71.2 ± 0.3**
100K	71.5 ± 0.7	**71.6 ± 1.2**
200K	71.0 ± 0.5	**71.0 ± 0.6**

Table 5.2: Macro-averaged F1 (± standard deviation) on test set. The C-GRU classifier is trained using only the generated and filtered data. The macro-average F1 score higher than the baseline results are indicated in bold.

slightly less than the performance of the classifier trained using only the original training set (baseline results). This shows that the quality of the generated data was similar to the original data. However, for the Davidson dataset, we note that the performance is higher than the baseline when more than 25K generated samples are used for training the C-GRU classifier. Compared to the baseline results, we observe a relative improvement of up to 4.2% (68.7 to 71.6) by using only the generated data for the classification. These results indicate that the generated data resembles the characteristics of the original training data and can be efficiently used for data augmentation. In comparison to the Founta training set, the Davidson training set has a fewer number of samples. Thus, the better results obtained using the model trained using only the generated data for the Davidson dataset show that our data augmentation can be more efficient when the number of samples in the training set is low.

5.4.3 Influence of filtering the generated samples

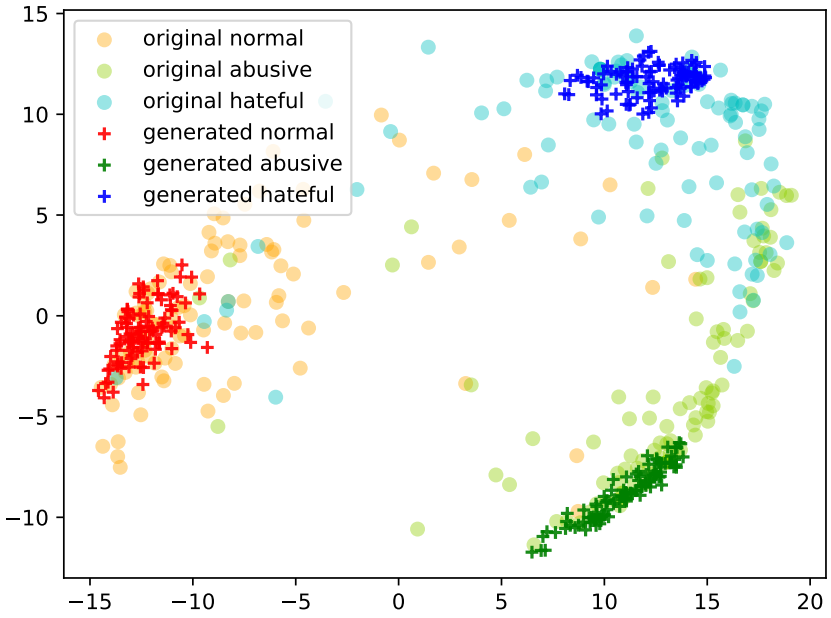
In this subsection, we explore the influence of using the BERT model for filtering the generated data. We use the top-N generated samples for the augmentation and compare against using the N generated samples randomly chosen for the data augmentation.
Table 5.3 shows the effect of using a fine-tuned BERT model for filtering the samples generated by GPT-2 for data augmentation on the Founta and the Davidson dataset. Here, for data augmentation, we randomly choose N-generated samples and compared them against the top-N samples sorted by the fine-tuned BERT model. We observe that choosing the samples filtered by the fine-tuned BERT model over the randomly chosen samples for augmentation gives a relative improvement of up to 2.1% (74.5 to 76.1) and 3% (70.7 to 72.8) for the Founta and the Davidson dataset, respectively. This shows that using the BERT-based sample filtering allows relevant samples from the generated data to be selected for the data augmentation.

	Founta test set		Davidson test set	
Amount of generated data used for each class	Random Sampling	Top-N scored by BERT	Random Sampling	Top-N scored by BERT
Baseline (no data augmentation)	73.5 ± 1.0		68.7 ± 0.7	
5K	73.5 ± 0.1	74.4 ± 0.3	71.6 ± 1.5	71.7 ± 0.6
25K	74.2 ± 0.6	75.3 ± 0.4	70.7 ± 1.8	72.8 ± 0.9
50K	74.5 ± 0.2	**76.1 ± 0.2**	71.7 ± 0.7	**73.2 ± 0.9**

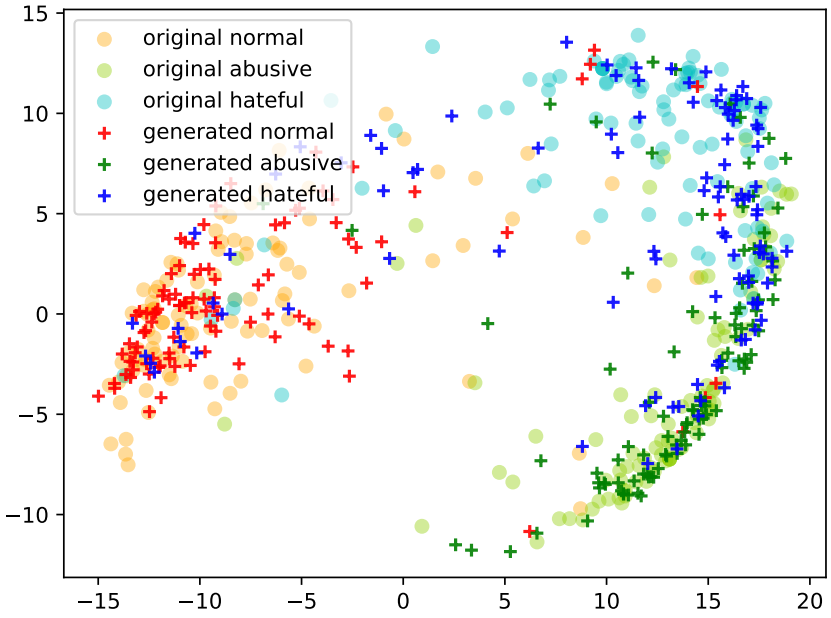
Table 5.3: Comparison of classification performance by augmenting N randomly sampled data versus top-N filtered by BERT. Macro-averaged F1 (± standard deviation) on Founta and Davidson test sets.

Additionally, we qualitatively evaluate the influence of filtering using the Principal Component Analysis (PCA) (Karl Pearson F.R.S., 1901) plots. We compare the PCA plot of the top 50K generated data sorted by the BERT model against the PCA plot of randomly chosen generated data used for the augmentation. To perform this, we use the fine-tuned BERT model for filtering. We use the embeddings of the '[CLS]' token as the model's representation for a given input sentence, the dimension of this embedding is 768. To visualize the embedding using a 2-dimensional plot, we perform the dimensionality reduction using the PCA. We show the PCA plots for our experiments on the Founta dataset.

Figure 5.4a shows the PCA scattered plot of the original training samples with the top-50K generated samples filtered using the fine-tuned BERT model. Figure 5.4b shows the PCA scattered plot of the original training samples with the randomly chosen generated samples. In Figure 5.4, the original training data belonging to 'normal', 'abusive', and 'hateful' classes are represented using orange, light green, and cyan colors, respectively. Similarly, the generated data belonging to 'normal', 'abusive', and 'hateful' classes are represented using red, green, and blue colors, respectively. The original training data is indicated using circles, and the generated samples are indicated using the '+' symbol. We observe that the data points belonging to a given class of the original training set are well separated from the data belonging to the other classes of the training set, with a few of the samples from a given class sparsely overlapping with samples from the other classes. The overlap is bigger for data belonging to the 'abusive' and 'hateful' classes. This indicates more similarity between the representation of the 'abusive' class and the 'hateful' class. In Figure 5.4a, we observe that the top 50K generated samples filtered by the BERT model appear as dense clusters, away from the data points of other classes and around the centroid of data points representing the original training data of the respective class. However, in Figure 5.4b, we observe that randomly chosen generated data overlap more with the data points from the other class when compared with the top 50K generated samples. This shows that choosing the randomly sampled generated data may not select the samples that strongly represent the target class. Henceforth, using a



(a) original training data and top 50K generated data filtered by the fine-tuned BERT



(b) original training data and randomly sampled generated data

Figure 5.4: PCA plots of original training data and GPT-2 model-generated data for the Founta dataset. Comparison of top-50K generated samples filtered by BERT with randomly sampled generated data. Embeddings from the fine-tuned BERT model is used to represent the data.

classifier-based filtering mechanism helps to choose the reliable generated data near the centroid of the original training set's target class.

Furthermore, to observe the influence of filtering, we qualitatively analyze the samples generated by the GPT-2 model and filtered by the fine-tuned BERT model. Table 5.4 present some of the representative examples of generated samples for the 'normal', 'abusive, and 'hateful' classes, respectively. The Founta dataset is used. We present the top-ranked and the bottom-ranked generated samples in the data sorted by the BERT model. We observe that the generated tweets are comprehensible to a human reader. We observe that the top-ranked samples belong to the desired target class, with the top-ranked 'normal' class samples not containing any profane words or intended hate. The top-ranked 'abusive' class samples contain profane words such as 'f*ck', 'bi*tch', etc. The top-ranked 'hateful' class samples contain racial slurs, targeted hate, etc. However, we observe that the bottom-ranked samples do not belong to the desired target class. This could be due to fine-tuning of the class conditioned GPT-2 model on samples from all three classes, hence, the GPT-2 model may generate few samples not belonging to the desired class. By generating a large number of samples, and selecting the top-N samples for data augmentation, the bottom-ranked samples were filtered out and not used to train the C-GRU classifier. This shows that BERT filtering performs a powerful selection of relevant samples from the generated data.

5.4.4 Evaluation of data augmentation in low-resource scenarios

To study the influence of data augmentation in the scenarios where the number of training samples is small, we simulate the low-resource set up by reducing the amount of original training data (a) to fine-tune the GPT-2 model for the data generation, (b) to fine-tune the BERT model for filtering, and (c) to train the C-GRU classifier. We conduct our experiments using 1000, 2000, 5000, 10000 training samples. However, for all the experiments presented in this section, we augment the amount of selected original training data with top 50K generated samples for each class. For example, when we consider a low-resource setting with 1000 training samples, we fine-tune the GPT-2 model with only 1000 training samples, then we generated 600K samples for each class. Further, we consider the same 1000 samples to fine-tune the BERT model, wherein the 'normal' and 'abusive' class samples are further down-sampled to have an equal amount of samples for all classes (as mentioned in Section 5.2.2.2). We then filter the generated data using the fine-tuned BERT model. We consider the same 1000 of the original training samples augmented by the top 50K generated samples for each class to train our C-GRU classifier. Figure 5.5a and Figure 5.5b show the macro-averaged F1 obtained for the low-resource settings on the Founta and the Davidson test sets respectively. We observe that the performance of the C-GRU model trained using only original training data, referred to as 'no augmentation' in Figure 5.5 improves with the increase in the amount of original training data used to train the C-GRU model. Further, we observe that augmenting 50K samples to each class, referred to as 'augment each class 50K' in Figure 5.5, performs similar or better than not augmenting the data for all the simulated low-resource scenarios for both

Table 5.4: Examples of high-scored and low-scored samples generated by the GPT-2 model trained on the Founta dataset, sorted by the BERT model.

(a) 'normal' class

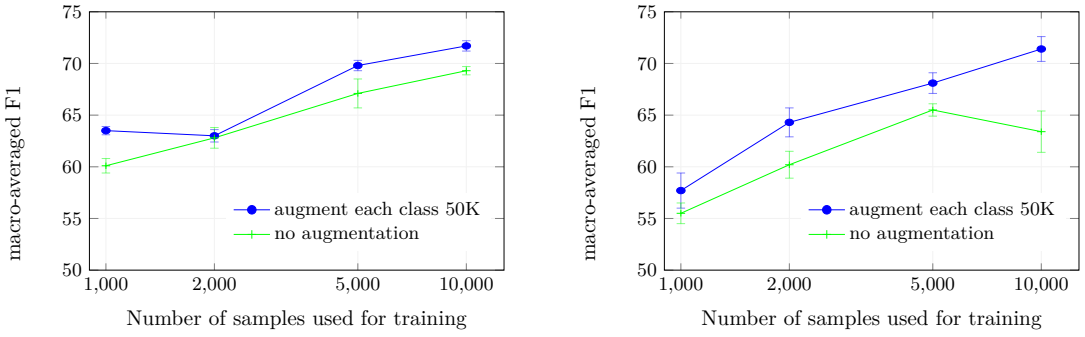
Top-ranked generated samples for the 'normal' class
a lovely sunday afternoon spent enjoying the sights, sounds and lovely people of burke bay. quality coast...
join us for lunch on the farm this sunday at our in - person pickup time! we are excited to be hosting this great.. on -
another beautiful day to reflect and enjoy the beautiful day ahead of us in delaware. i'll be hanging out on... leophotography...
Bottom-ranked generated samples for the 'normal' class
@user when u get on his nerves but he cant stop fucking with you even if he tried he probably wouldve lost his shit at that point anyway
@user no one fucks with dems. no matter what is the protocol, the con... that is not the protocol. everyone fucks with
@user god i fucking hate i hate ppl like that! i don't give two fucks! but you're a pikin girl!

(b) 'abusive' class

Top-ranked generated samples for the 'abusive' class
@user i need a fucking rest, im all kinds of fucked up, can't even fall asleep anymore, cant even fall asleep at am, im
@user i need a bad bitch to takeover my snapchat so i can stop recording and looking at my shit lmaoo this shit is crazy me
wtf @user why do i keep getting returned calls for dm's? i dont have time to respond to that much shit! sms
Bottom-ranked generated samples for the 'abusive' class
@user can't thank my nxt family enough for everything they've done for me in the last year, it's truly blessed. no words
i don't understand people don't believe the insane weather forecast of kentucky. crazy hail, gusting up to mph.
i don't mind going out on dates, it's not a requirement but a preference. my preference is wild, passionate, and outdoorsy girls.

(c) 'hateful' class

Top-ranked generated samples for the 'hateful' class
@user this what happens when you separate yo self from niggas who don't eat they food cold. you flourish.
please tweet if you hate the black family why would you pay a mf millions to kill off a race? al qaeda like that's what u
i hate ppl who post pictures like this. @user shame on you. ppl like this should be lobotomized. very, very sad
Bottom-ranked generated samples for the 'hateful' class
@user its happening! a song. a video. a preorder lets fucking go now or never.
ive been tryna get an account on hitmontop since m s. just gonna wait till we get hitmontop x hitmontop and we
your support makes a big difference



(a) Founta (b) Davidson

Figure 5.5: Macro-averaged F1 (± standard deviation) on test set for low-resource setting. The C-GRU classifier is trained using varying amounts of original training data (X-axis) and 50K samples of generated tweets. The GPT-2 model and the BERT model are also fine-tuned with the varying amounts of original training data.

the Founta and the Davidson datasets. For the Founta dataset, we obtain up to 5.7% (60.1 to 63.5), 4.0% (67.1 to 69.8), and 3.5% (69.3 to 71.7) of relative improvement by using 1000, 5000, and 10000 training samples in low-resource setup respectively. Similarly, for the Davidson dataset, we obtain up to 4.0% (55.5 to 57.7), 6.8% (60.2 to 64.3), 4.0% (65.5 to 68.1), and 12.6% (63.4 to 71.4) of relative improvement by using 1000, 2000, 5000, and 10000 training samples. Our low-resource experiments demonstrate the effectiveness of our data augmentation approach when the amount of available training samples is low.

5.5 Conclusion

In this chapter, we explored data augmentation to improve hate speech classification. The data augmentation is performed by generating large number of samples from a GPT-2 model fine-tuned using the objective of conditional language modeling. The generated samples are then filtered using the BERT model to select high quality samples, which are then added to the original training data to train the C-GRU classifier.
We analysed the influence of quantity of data used for augmentation. Our experiments showed that adding a few thousand generated samples to the training set yields a significant gain in performance compared to the baseline approach of training the classifier only with the original training data. Furthermore, we also explored augmenting samples only to the minority 'hateful', and showed significant improvements in the classification performance. Our results show that data augmentation using the synthetic samples generated by the GPT-2 model helps to improve hate speech classification performance. Our approach of data augmentation using a single class conditional language model for data generation provided slightly better results compared to using multiple GPT-2 models to generate the data for different class labels.

We analyzed the quality of the generated data by evaluating classifiers trained only on the generated data. The results showed that a model trained with a sufficient amount of generated data can surpass the performance of the model trained using only the original training data. This shows that the data generated using language model well captured the features of the original training set, and can be efficiently used for the data augmentation. We investigated the influence of using the fine-tuned BERT model to filter the generated data and showed that using BERT-based filtering helps to choose pertinent samples for data augmentation. Furthermore, using the PCA plots, we show the ability of the BERT model to filter out the generated samples overlapping with the samples belonging to the non-target class.
Finally, we simulated a low-resource setting by reducing the amount of available training data. We showed that our approach of data augmentation provides performance gain even when we have very few thousand labeled training samples. Thus, we can employ a language model-based data augmentation technique to improve the hate speech classification performance when the number of training samples is small.

6 Multi-corpus learning for hate speech classification

6.1 Introduction

In this chapter, we adapt the paradigm of multi-task learning to multi-corpus learning for hate speech classification. Multi-task learning is motivated by the capabilities of humans to simultaneously learn multiple related tasks together. For example, a person learning to play two musical instruments like a piano and a guitar can transfer some knowledge of music learned in piano for learning the guitar and vice-versa. In neural networks, multi-task learning aims to optimize a model by jointly learning multiple related tasks. Thus, combining the domain-specific information of related tasks to benefit the generalization of the model for all the tasks (Caruana, 1997). This chapter aims to harness different hate speech corpora in a multi-task learning setup by associating one task to one corpus. This multi-corpus approach is expected to improve classification performance and domain adaptation capabilities of the model.

Multi-task learning has applications in various domains, such as computer vision, bioinformatics, speech, Natural Language Processing (NLP), etc (Zhang and Yang, 2021; Ruder, 2017). In NLP, multi-task learning aims to jointly learn tasks such as Named Entity Recognition (NER), sequence classification, Part-Of-Speech Tagging (POS), etc. (Collobert and Weston, 2008; Ruder et al., 2019). For a supervised learning task, such as classification, a deep neural network-based classifier's performance depends on the number of labeled samples available to train the system. Multi-task learning leverages the advantage of combining the datasets of related tasks to jointly train the model, thereby creating an implicit data augmentation and reduces the data sparsity issue (Ruder, 2017; Worsham and Kalita, 2020). Multi-task learning and language modeling are specific types of transfer learning. In language model pre-training, the model aims to learn the generic representation of the language using large unsupervised data (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2018), that can be helpful when the model is fine-tuned for a new task. However, multi-task learning aims to jointly learn the related supervised learning tasks, thus combining domain-specific information, reducing data dependent noise, and improving shared representation between the related tasks for the benefit some or all the tasks. The objective of language modeling and multi-task learning is combined in the works of Liu et al. (2019b) to take advantage of both approaches. Thus, in this chapter, we explore the multi-task learning approach using a pre-trained language model.

6.1.1 Diversity of hate speech corpora

Hate speech datasets are collected from various sources such as Twitter (Davidson et al., 2017; Founta et al., 2018; Basile et al., 2019), Wikipedia (Wulczyn et al., 2017), Reddit (Qian et al., 2019), etc. The style of the comments largely varies from one source to another. For example, the comments on Twitter are generally shorter in comparison with the comments on Wikipedia talks. On one hand, authors of some datasets annotate for binary classification tasks with labels such as 'toxic speech' and 'non-toxic speech', which covers all forms of hate terms under a single umbrella term 'toxic speech'. While, on the other hand, some authors create more challenging datasets by using fine-grained labels such as 'abusive speech', 'hate speech', 'racism', 'sexism', etc. Furthermore, the sampling strategy, the time frame (Florio et al., 2020) during which the comments are collected, and the definition of class labels (Kapil and Ekbal, 2020; Fortuna et al., 2020) used by annotators add to the diversity between various datasets. The diversity among the available hate speech corpora makes a model trained on one dataset inefficient for testing the samples from another dataset (Wiegand et al., 2019), thus reducing generalization. The difference among these datasets also prevent them from directly combining them. Although these datasets individually differ from each other, a commonality between them is that they contain hate speech. By using multi-task learning approach, a single model can be jointly trained on several diverse hate speech corpora, thus improving the shared representations learned by the model from one corpus to benefit for another corpus.

6.1.2 Multi-task learning in hate speech classification

Multi-task learning is also explored for hate speech classification by Waseem et al. (2018); Kapil and Ekbal (2020); Plaza-Del-Arco et al. (2021); Rajamanickam et al. (2020). Kapil and Ekbal (2020) explored different deep learning models such as Convolutional Neural Networks (CNN) (Kim, 2014), Long Short-Term Memory(LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unit (GRU) (Cho et al., 2014) based models for the multi-task learning architecture. The authors combined the features obtained from the trained multi-task learning model with the single-task learning model, thus combining shared features and task-specific features to improve hate speech classification tasks. Rajamanickam et al. (2020) used Bidirectional-LSTM based multi-task learning architecture, wherein, the features of single-task learning were combined with multi-task learning using an attention mechanism. Furthermore, the authors used emotion detection as an auxiliary task along to improve classification results on two hate speech classification tasks. Though these prior works showed the effectiveness of multi-task learning architectures for hate speech classification, they failed to exploit the availability of pre-trained models for the multi-task learning of hate speech classification. Compared to the works of Kapil and Ekbal (2020); Rajamanickam et al. (2020) we incorporate the pre-trained Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) model for our multi-task learning to benefit from extensive knowledge learned by the BERT per-training.

The multi-task learning approach explored in this chapter is based on the work of Liu

et al. (2019b). The authors combined a range of NLP tasks using the shared layers represented by the pre-trained BERT model and several groups of task-specif layers, each group corresponding to a single-task. Similarly, we use the pre-trained BERT model as our shared hidden layers for multi-task learning architecture. Compared to this work, we adapt the paradigm of multi-task learning to multi-corpus learning. In our approach, we treat each dataset as a classification task, hence, one task corresponds to one corpus. Architecture based on pre-trained model is explored by Plaza-Del-Arco et al. (2021), wherein, the BERT model pre-trained on Spanish (Canete et al., 2020) for multi-task learning is used for Spanish datasets, the authors showed improvements in hate speech classification by using sentiment analysis and emotion analysis datasets along with the two hate speech classification datasets as auxiliary tasks. However, unlike (Plaza-Del-Arco et al., 2021), we do not incorporate sentiment analysis and emotion analysis tasks. Instead, we exploit the relatedness of hate speech classification tasks by using five hate speech datasets, based on Tweets (Davidson et al., 2017; Founta et al., 2018; Wulczyn et al., 2017; Basile et al., 2019) and the Wikipedia talks user comments (Wulczyn et al., 2017).
Furthermore, we explore low-resource domain adaptation for hate speech datasets based on language model adaptive-pretraining (Ramponi and Plank, 2020) using the multi-corpus learning approach.

To summarize, we present the following contributions:

1. We adapt multi-task learning approach to multi-corpus learning for hate speech classification using widely used hate speech corpora.
2. We simulate low-resource scenarios to study the effectiveness of multi-corpus learning when the number of training samples are small.
3. We study low-resource domain adaptation for hate speech classification using multi-corpus learning.

The following contents of this chapter are organized as follows, section 6.2 describes the objective of multi-task learning, architectural differences between multi-corpus and single-corpus learning, and domain adaptation. Section 6.3 describes our approach for multi-corpus learning using a pre-trained model. Section 6.4 describes the datasets and model parameters used for our multi-corpus learning approach. The results obtained for our multi-corpus learning are discussed in Section 6.5.

6.2 Multi-task learning

In this section, we briefly describe the objective of multi-task learning, the architectural comparison of single-task-learning and multi-task learning.

6.2.1 Objective of multi-task learning

Given T tasks $\{t_i\}_{i=1}^{T}$, where all or some of the tasks are related, multi-task learning aims to jointly learn T tasks to improve the model performance on every task t_i by leveraging the information contained in all the T tasks.
Let us consider a dataset D_i corresponding to the supervised learning task t_i, having n_i number of samples of form $\{x_j^i, y_j^i\}_{j=1}^{n_i}$, where x_j^i is the input feature, and y_j^i is the target of the sample j belonging to task t_i. For a given task t_i, a multi-task learning model learns the parameter set $\{\theta^s, \theta^i\}$ using a function f^i as follows:

$$f^i(X^i; \theta^s, \theta^i) : X^i \rightarrow Y^i \tag{6.1}$$

where $(x_1^i, x_2^i \dots x_{n_i}^i) \in X^i$, $(y_1^i, y_2^i \dots y_{n_i}^i) \in Y^i$, θ^s is the model parameters shared between all the tasks in $\{t_i\}_{i=1}^{T}$, and θ^i is the task-specific model parameters for only the task t_i. The objective of the model is to minimize the overall loss L as:

$$L(\theta^s, \theta^1, \theta^2, \dots \theta^T) = \sum_{i=1}^{T} L^i(\theta^s, \theta^i) \tag{6.2}$$

where $L^i(\theta^s, \theta^i)$ is the loss for task t_i. For a supervised learning task t_i, the loss L^i corresponds to:

$$L^i(\theta^s, \theta^i) = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathcal{L}(f^i(x_j; \theta^s, \theta^i), y_j) \tag{6.3}$$

Where, $\mathcal{L}$ is a loss function measuring how well the function f^i fits the training data (X^i, Y^i). The objective of multi-task learning is to reduce the overall loss L, by optimizing the parameters shared across all the tasks θ^s and task-specific parameters $\{\theta^i\}_{i=1}^{T}$.
In a single-task learning approach, $T = 1$ and the dataset of task t_1 is processed by a model with parameters θ^1.

6.2.2 Architecture of multi-task learning

Multi-task learning can be achieved using two approaches, using hard parameter sharing and soft parameter sharing. A generic block diagram of the single-task learning architecture and its comparison with the multi-task learning architecture using hard parameter sharing and soft parameter sharing is shown in Figure 6.1, Figure 6.2, and 6.3, respectively.
In a single-task learning approach, each dataset D_i corresponding to task t_i is processed by a set of task-specific layers having parameters θ^i, using the data for the given task t_i. These task-specific layers do not share the parameters with other tasks. But in the case of the multi-task learning setup using hard parameter sharing, all the datasets $\{D_i\}_{i=1}^{T}$ are first processed by shared layers having the learnable parameters θ^s. Thus, the shared layers benefit by learning a shared representation for all the tasks based on relevant information captured from all the available input data from every task. The output of

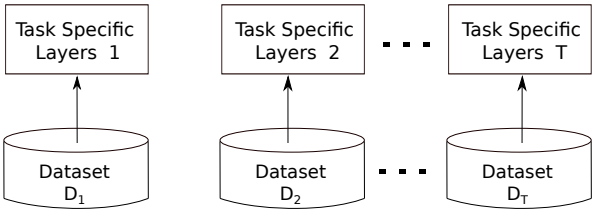


Figure 6.1: Block diagram of single-task learning.

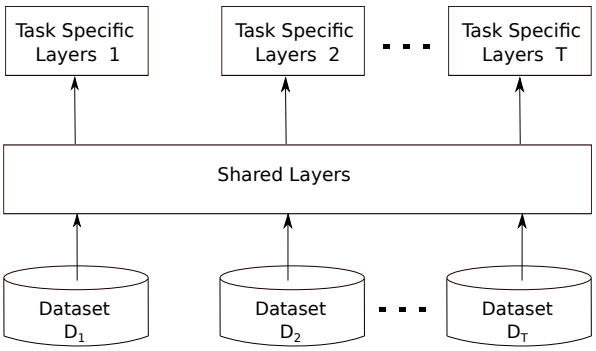


Figure 6.2: Block diagram of multi-task task learning using hard parameter sharing.

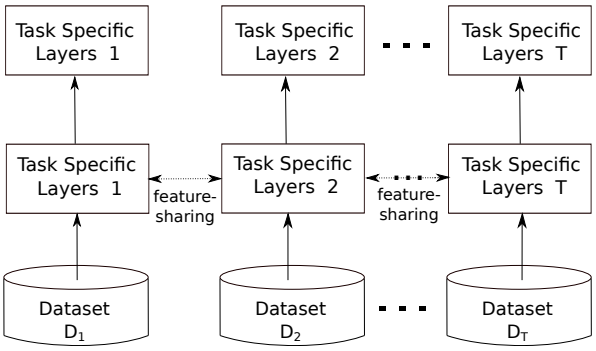


Figure 6.3: Block diagram of multi-task task learning using soft parameter sharing.

the shared layers is then passed as an input to the task-specific layers having parameters θ^i for a given task t_i. However, in case of multi-task learning using soft parameter sharing, the dataset D_i is processed by the set of task specific layers t_i, and then some of these task-specific layers are regularized during training to reduce the differences between their layer parameters (Worsham and Kalita, 2020; Ruder, 2017).

6.3 Proposed methodology

In this section, we describe our proposed methodology for multi-corpus learning using the method of multi-task learning. We also describe the domain adaptation for hate speech

classification using multi-corpus learning.

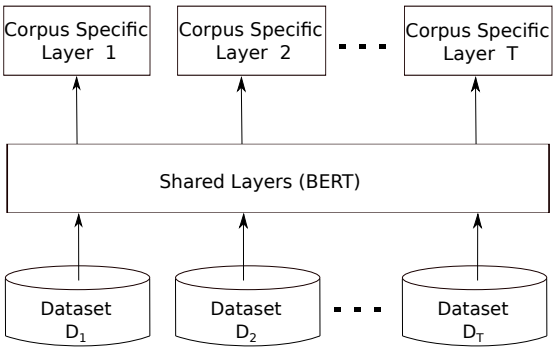


Figure 6.4: Methodology adopted for multi-corpus learning of hate speech classification.

6.3.1 Multi-corpus learning using pre-trained language model

Our methodology is based on multi-task learning using hard parameter sharing, adapted from the works of Liu et al. (2019b), wherein, we use the pre-trained BERT model as shared layers. Furthermore, we treat one corpus as one task, thus, adapting the paradigm of multi-task learning to multi-corpus learning. Figure 6.4 shows the block diagram of our approach of multi-corpus learning for hate speech classification. The multi-corpus learning architecture comprises two parts, (a) shared layers; and (b) a set of corpus-specific layers. The number of corpus-specific layers is equal to the number of datasets used to train the model.

Shared layers: The shared layers are the initial layers of the multi-corpus model and are shared by all the datasets. To leverage the representational capabilities of a pre-trained model, we use the transformer-based (Vaswani et al., 2017) pre-trained Bidirectional Encoder Representations from Transformers (BERT) model as our shared layers. First, the input sentence is split into a sequence of word-piece tokens based on the pre-trained vocabulary of the BERT model. The sequence of word-piece tokens is then passed as an input to the BERT model. The output of the BERT model is the sequence of the contextual embedding representations. The training samples from all the tasks are passed as input to the shared layers. Therefore, the model parameters of the shared layers are updated using all training data of each corpus. Henceforth, the shared layers also benefits from an implicit data augmentation as it processes the combined data from all the datasets. This also enriches the representations learned by the shared layers based on the relevant datasets.

Corpus-specific layers: We use one set of corpus-specific layers for each corpus. The output of the shared layers is used as an input to the corresponding corpus-specific layer. The objective of the corpus-specific layers is to optimize the model for a given corpus. Since all the considered datasets for our multi-corpus approach are classification tasks, the corpus-specific layers are used to perform either binary classification or multi-class classification. The corpus-specific layer parameters are updated only when the data from

the corresponding corpus is used as an input to the model.

6.3.2 Domain adaptation using multi-corpus learning

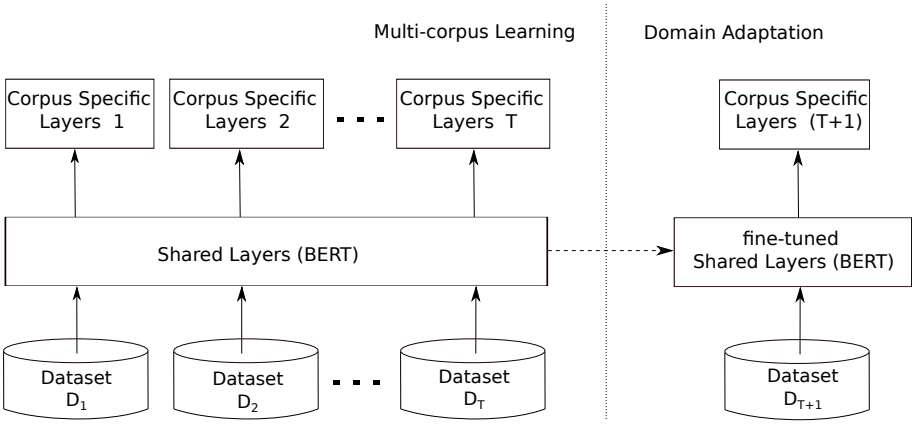


Figure 6.5: Block diagram for domain adaptation using multi-corpus learning.

In a supervised learning setup, a model is trained using the training data and tested on unseen data. Generally, it is assumed that the data of the training set and the test set come from the same distribution (domain). However, when the distribution of train and test sets differ, it is referred to as domain shift (Ramponi and Plank, 2020). The domain shift reduces ability of the model trained on one domain to adapt to another domain. The domain adaptation technique allows a model to adapt to target domain having a limited amount of data. In supervised domain adaptation, a few amounts of labeled data of the target domain are available. As we have the labeled data, we perform supervised domain adaptation using multi-corpus learning architecture.
We observe that the hate speech datasets differ in terms of the definition of the class label, the time frame of data collection, sampling strategy of the dataset, etc. (Kapil and Ekbal, 2020) This creates a domain shift among the available datasets. Henceforth, we use multi-corpus learning as an approach for domain adaptation technique for hate speech classification.
The block diagram for domain adaptation using multi-corpus learning approach is presented in Figure 6.5. To perform domain adaptation on a target corpus using the multi-corpus learning approach, we first train a multi-corpus model with all the available tasks except one, which is our target corpus for the domain adaptation. Then we adapt the trained multi-corpus learning model to our target corpus. Our method for domain adaptation is as follows: we add a new corpus-specific layers for the target corpus, then we train the shared layers together with the added corpus-specific layers. This adapts the shared layers of the model trained on multiple corpora to the target corpus, as well as learns new corpus-specific layers for the target task.

6.4 Experimental setup

In this section, we briefly describe the considered datasets, and the choice of model parameters for our multi-corpus learning approach.

6.4.1 Datasets

We consider five widely used hate speech datasets to train our multi-corpus learning model. Four of these datasets are tweets sampled from the Twitter, namely 'Davidson' (Davidson et al., 2017), 'Founta' (Founta et al., 2018), 'Hateval' (Basile et al., 2019), and 'Waseem' (Waseem and Hovy, 2016). The fifth dataset is sampled from Wikipedia talk pages (Wulczyn et al., 2017), further in this chapter we refer to it as 'Wikipedia'. The Davidson and Founta datasets are used for the multi-class classification of hate speech. Whereas, we perform binary classification for the Hateval, Waseem and Wikipedia datasets. A detailed explanation of the datasets is provided in Section 3.5. The text pre-processing used for these datasets are provided in Section 3.6.

6.4.1.1 Dataset split

For Davidson, Founta, and Waseem, we randomly split the datasets into three portions- 'training', 'validation', and 'test' sets, each containing 70%, 10%, and 20% respectively. For Hateval and Wikipedia datasets, we use the splits provided by the datasets. We use the 'training' set to train the model parameters, the 'validation' set is used to tune the model parameters and for early stopping, and the 'test' set to evaluate and report the final model performance.

6.4.2 Multi-corpus model and training description

We use the pre-trained 'bert-base-uncased' model trained on English corpus as our shared layers. This model has 110M trainable parameters, and the embedding dimension is 768. We use the implementation of Huggingface's transformers API (Wolf et al., 2020) to incorporate the BERT model for our implementation. The BERT model outputs a sequence of contextual embeddings where the output embeddings of any given token are based on its word-piece token and all tokens from the input sentence (context). Furthermore, the output embeddings provided by the shared model are the shared representations of all the tasks.
As we use five datasets, and all the tasks are classification tasks, we use five task-specific layers. The output of the last hidden state corresponding to the '[CLS]' token from the shared layers can be considered as an aggregate output representation of the entire input sentence, hence, used as an input to the corpus-specific layers. As the embedding dimension of the BERT model is 768, we use a single Dense layer with 768 hidden units as our corpus-specific layer. The outputs of this hidden layer are passed through a output layer with softmax activation. The number of units in output layer is equal to the number of classes for the respective corpus.

6.4.2.1 Training and model parameters

Compared to the standard way of a random selection of samples for a mini-batch for training, we perform a corpus-specific selection of mini-batches. Our method of mini-batch selection is as follows: given two datasets, we select a fixed number of random training samples from the first dataset as the first mini-batch, we select the same number of samples from the second dataset as the second mini-batch. For the third mini-batch, we select the fixed number of random samples from the first dataset, and the process continues until all the samples from the largest training set are used for training. When one dataset has fewer samples compared to another dataset, the samples from the smaller dataset are repetitively selected. This kind of mini-batch selection ensures that the multi-corpus model is trained with an equal number of samples from all the datasets, and prevents getting biased to the larger dataset. For example, if the two datasets D_1 and D_2 have 3200 and 1600 training samples, respectively, and the mini-batch size is 32, we have 100 mini-batches for the dataset D_1 for an epoch of training. However, since we use an equal number of mini-batches for all the datasets, we have 100 mini-batches for dataset D_2 as well. However, 100 mini-batches for the dataset D_2 indicates 3200 samples, thus selecting 1600 samples twice.
We use ReLU (Agarap, 2018) activation for the dense layer of task-specific layers. We use an initial learning rate of 1×10^{-5}, Adam (Kingma and Ba, 2015) optimizer, a maximum of 30 epochs, and early stopping based on the validation set. At the end of each epoch, the performance of the validation set is evaluated using macro-averaged F1-measure averaged across all five tasks. We choose the model checkpoint that performs the best on the validation set and report the results using the test set.

6.5 Results and discussion

We use macro-averaged F1-measure as our metric to report the performance on the test set. We report the mean and standard deviation evaluated over five runs of the model with different random initializations. The 95% confidence interval on macro-averaged F1 obtained using paired bootstrap (Dror et al., 2018; Efron and Tibshirani, 1994) is ±2.7, ±1.5, ±2.5, ±2.5, and ±0.8 for Davidson, Founta, Hateval, Waseem, and Wikipedia test sets, respectively. In the following of this section, we compare the results obtained on the multi-corpus approach with the single-corpus approach, discuss the results obtained by combining a few corpora into a single-corpus, discuss the results on the low-resource domain adaptation using the multi-corpus learning approach.

6.5.1 Multi-corpus learning

We have compared the results of three setups, Single-Corpus Learning (SCL), Multi-Corpus Learning (MCL), and Multi-Corpus Learning with corpus-specific fine-tuning ($\text{MCL}_{\textit{finetuned}}$). These setups are explained below.

6.5.1.1 Single-Corpus Learning (SCL)

We establish our baseline results using the SCL approach. The SCL approach involves fine-tuning the pre-trained BERT model on a single dataset using a single set of corpus-specific layers. It involves updating the pre-trained model parameters of the BERT model along with a dense layer for the classification task. This model differs from the multi-corpus model in the fact that the SCL model is trained using only a single dataset and does not benefit the usage of multiple corpora.

6.5.1.2 Multi-Corpus Learning (MCL)

For our MCL, we use the model described in section 6.4.2, where the shared layers use the pre-trained BERT model and five sets of corpus-specific layers for five datasets. This model is jointly trained using five datasets. The model that gives the best average results on all five corpora is considered for evaluating the performance on the test set.

6.5.1.3 Multi-Corpus Learning with corpus-specific fine-tuning ($\text{MCL}_{\textit{finetuned}}$)

The MCL model can be further optimized on a target corpus. Thus, we consider the trained MCL model and further fine-tune it by using a single corpus. Thus, it is a two-step training, where the first step involves jointly training a model using multiple hate speech corpora (MCL setup). Then the second step involves further fine-tuning the model obtained in the first step with the objective of single-corpus learning (SCL). This updates the parameters of the shared layers and the corpus-specific layers for a single target corpus.

The results obtained on the test set for the five considered datasets using the 'SCL', 'MCL', and '$\text{MCL}_{\textit{finetuned}}$' setup are shown in Table 6.1. We have also reported the average performance of the these setups.

Table 6.1: Macro-averaged F1 (± standard deviation) results on the test set for the multi-corpus learning approach.

	Davidson	Founta	Hateval	Waseem	Wikipedia	Average
SCL	76.0 ± 0.6	75.8 ± 0.4	49.3 ± 1.8	84.0 ± 0.5	86.9 ± 0.1	74.4
MCL	76.3 ± 1.1	75.5 ± 0.2	50.4 ± 3.0	84.1 ± 0.4	86.4 ± 0.2	74.5
$\text{MCL}_{\textit{finetuned}}$	75.7 ± 1.0	75.8 ± 0.7	52.1 ± 2.6	84.6 ± 0.6	86.7 ± 0.2	**75.0**

From Table 6.1, we observe that the average macro-averaged F1-results obtained on five datasets for our baseline SCL approach is 74.4. Wherein, the F1 scores obtained on the Waseem and Wikipedia datasets are higher than the results obtained for Davidson and Founta datasets.
We observe that the average result obtained using the MCL approach is 74.5, which is close to the average result obtained by the SCL approach. We note that Davidson, Hateval, and Waseem datasets with smaller training set have slightly benefited from the

MCL approach, however, Founta and Wikipedia with larger training set have shown marginally reduced performance. This shows that the smaller datasets benefit from the knowledge transferred from the larger datasets.
For the MCL$_{finetuned}$ setup, we obtained an average F1 of 75.0. We observe that all the datasets except Davidson have benefited from fine-tuning the MCL model. This shows that the MCL model benefited from further fine-tuning on a single corpus. Moreover, this also shows that the MCL model was not fully optimized for every considered task, as it was jointly fine-tuned using multiple tasks. Additionally, we show that the shared representations captured by the MCL model help when we fine-tune the model further with a related downstream task.
We have obtained very poor classification results on Hateval test set for all the setups, however, we have obtained better results on the Hateval validation set of 78.5±0.5 for SCL setup. Moreover, our obtained result on the test set for all the setups are higher than the average results obtained by the participants of the 'SemEval-2019 Task 5' challenge (Basile et al., 2019), wherein, the mean macro-averaged F1 obtained by the participants is 44.84. Furthermore, as mentioned by the providers of dataset (Basile et al., 2019), our results for the Hateval dataset follows a similar pattern, where approximately 80% of the 'non-hateful' samples in the test set are misclassified as 'hateful'.

6.5.1.4 Multi-corpus learning in low-resource scenarios

To analyze the performance of multi-corpus learning when the amount of available training data is small, we further explore the multi-corpus learning approach in low-resource scenarios. Our experiments follow the procedure of down-sampling the available training set of all the considered corpora to 100, 200, 500 and 1000 samples. We then perform the experiments for the SCL, MCL, and MCL$_{finetuned}$ approaches on the reduced training data. Table 6.2 presents the average macro-F1 on the five corpora in low-resource scenarios. Figure 6.6a, 6.6b, 6.6c, 6.6d, and 6.6e show the macro-averaged F1 results for SCL, MCL, and MCL$_{finetuned}$ setup for Davidson, Founta, Hateval, Waseem, and Wikipedia test sets, respectively.

Approaches	Number of training samples			
	100	200	500	1000
SCL	49.4	53.4	67.7	70.2
MCL	57.7	63.3	67.0	69.6
MCL$_{finetuned}$	**61.2**	**65.3**	**68.7**	**70.6**

Table 6.2: Average macro-F1 results on test sets for five corpora in low-resource scenarios.

From Table 6.2 and Figure 6.6, we can note that MCL and MCL$_{finetuned}$ setups show similar or better results than SCL for all the cases of the varying amount of training data. However, we observe that MCL and MCL$_{finetuned}$ give significant performance gains in very low-resource scenarios.
When we use 100 samples for the training sets, we obtain a relative improvement of

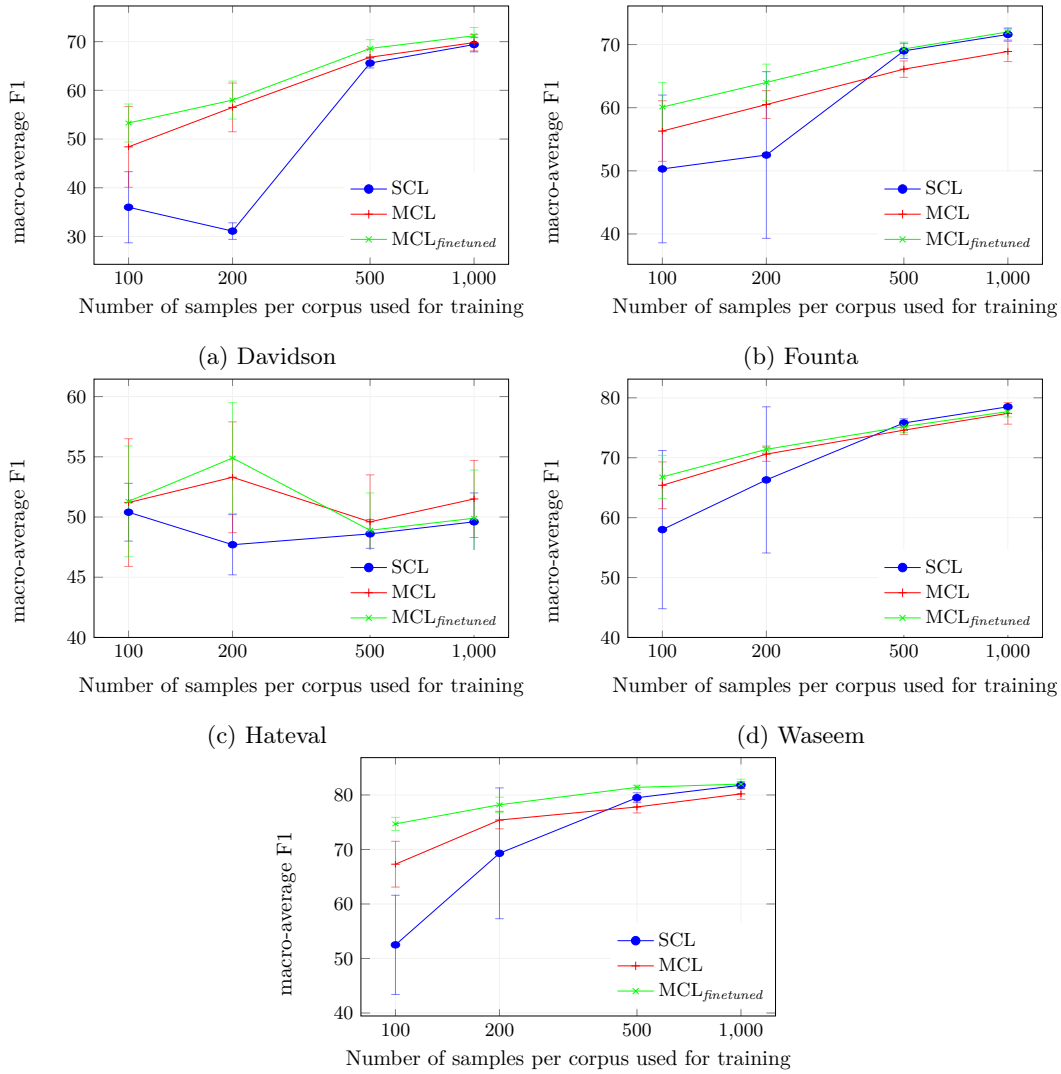


Figure 6.6: Macro-averaged F1 results on test set for low-resource scenario.

16.8% and 23.9% for MCL and MCL$_{finetuned}$, respectively, compared to SCL (49.4 to 61.2 and 57.7). When we utilize 200 samples for the training sets, we obtain a relative improvement of 18.5% and 22.3% for MCL and MCL$_{finetuned}$, respectively, compared to the SCL approach (53.4 to 65.3 and 63.3). These improvements are statistically significant.

These results indicate that when the number of available samples is low, the MCL model benefits from jointly training the model using several datasets. Furthermore, the results also show that corpus-specific fine-tuning of a model trained on MCL setup gives significant performance improvements compared to SCL setup in very low-resource scenarios.

These results show the usefulness of the multi-corpus learning approach when the amount of the training data is very low.

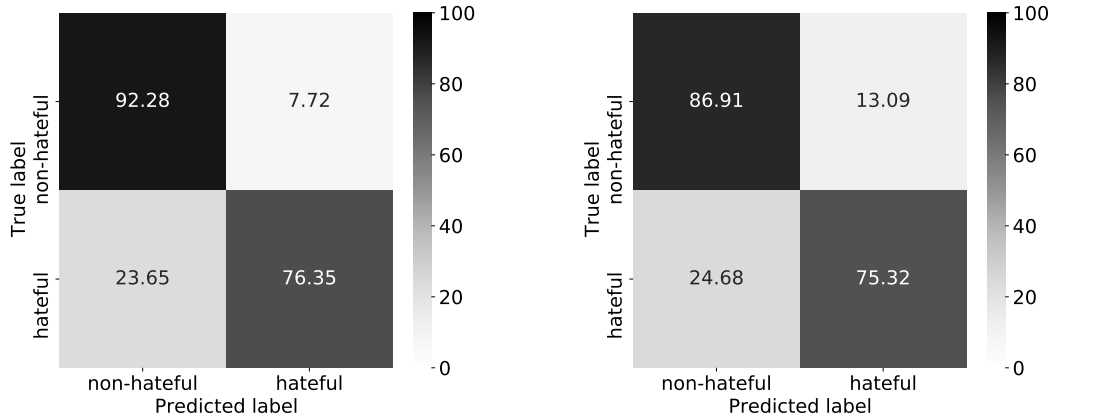
6.5.2 Combining tasks (corpora) for multi-corpus learning

Compared to the SCL approach, the MCL approach increases the number of samples to train the model by jointly training several datasets together. However, only the shared layers benefits of the MCL model benefit from jointly training with multiple datasets. Whereas, each set of corpus-specific layers will be trained with only the available corpus-specific data. Thus, to increase the amount of data used to train a corpus-specific layer, we combine the related corpora into a single one. In this work, we have combined the training sets of three-class classification tasks, i.e., Davidson and Founta datasets into a single training set, thus a single corpora. Likewise, we have combined the Hateval, Waseem, and Wikipedia training set corresponding to binary classification corpora into a single corpus. We have represented the combined training set as {Davidson & Founta} and {Hateval & Waseem & Wikipedia} in Table 6.3. This setup also reduces the number of corpus-specific layers used for the MCL architecture.
In this setup, for all the configurations, we have fine-tuned the models using the combined training set. However, we have evaluated the model by using the specific test set and have reported the results separately to allow the comparison with the previous results.

Train set	{Davidson & Founta}		{Hateval & Waseem & Wikipedia}			Average
Test set	Davidson	Founta	Hateval	Waseem	Wikipedia	
SCL	82.1 ± 4.7	77.2 ± 0.7	39.8 ± 2.3	80.4 ± 1.7	86.7 ± 0.2	73.2
MCL	82.2 ± 3.3	77.7 ± 1.1	42.5 ± 3.6	80.0 ± 1.2	86.4 ± 0.3	73.7
$MCL_{finetuned}$	88.1 ± 1.7	78.4 ± 0.2	42.3 ± 2.5	81.9 ± 0.7	86.0 ± 0.3	75.3

Table 6.3: Macro-averaged F1 (± standard deviation) results on the test set for the multi-corpus learning approach by combining tasks.

Table 6.3 presents the results obtained for SCL, MCL, and $MCL_{finetuned}$ setups using the multi-corpus learning approach by combining few datasets. For the SCL approach, we obtain better results for Davidson and Founta test sets compared to the SCL approach of not combining the training set (results in Table 6.1). This is probably because of similar definitions of the class labels used for the Davidson and Founta datasets. For the Wikipedia dataset, we obtain similar results as that of not combining the training set. However, we observe significantly reduced performance for Hateval and Waseem test sets. We analyzed these results using confusion matrices. Figure 6.7 compares the confusion matrices obtained on Waseem test set using Waseem training set and {Hateval & Waseem & Wikipedia} training set in SCL setup. Our analysis on the test set results for the model trained by combining the training set showed an increase in false positives for Hateval and Waseem test sets, where more samples from the 'non-hateful' class were misclassified as 'hateful' for the Hateval dataset. Similarly, more samples from the 'non-hateful' class were misclassified as 'sexism' for the Waseem dataset. This is probably because, samples with profane words belong to the 'toxic' class in Wikipedia dataset,



(a) Waseem training set

(b) {Hateval & Waseem & Wikipedia} training set

Figure 6.7: Confusion matrices on the Waseem test set. Comparison of using Waseem training set and {Hateval & Waseem & Wikipedia} training set.

whereas, such samples were considered as 'non-hateful' in Hateval and Waseem datasets. This creates a difference in labeling schemes used for Waseem, Hateval, and Wikipedia datasets. Thus, we show that combining similar corpora into a single corpus helps to better train the corpus-specific layers when the corpora use a similar class label definition. The MCL approach by combining the tasks gives macro-averaged F1 of 73.7. Furthermore, for $\text{MCL}_{\textit{finetuned}}$ approach, we obtain the best results (75.3), an average relative improvement of 2.9% and 2.2% over SCL approach and MCL approach, respectively. Additionally, we observe that Davidson, Founta, and Waseem dataset shows a significant performance gain compared to the SCL and MCL approaches. Compared to the MCL approach, we observe up to 7.2% (82.2 to 88.1), 1.6% (77.2 to 78.4), and 2.4% of relative improvements on Davidson, Founta, and Waseem datasets respectively. Thus, we note that corpus-specific fine-tuning of MCL model with combined corpora shows improvements compared to the MCL approach. Hence, showing that the shared representations learned by the MCL model are helpful for fine-tuning the model for related downstream tasks.

6.5.3 Domain adaptation using multi-corpus learning approach

We explore supervised domain adaptation for hate speech classification datasets using the multi-corpus learning approach. We perform this as follows: we train our MCL model using the training set of four of five available corpora. We consider the fifth corpus as the target corpus for domain adaptation. We then use the trained MCL model and add a new set of corpus-specific layers for the target corpus. We then fine-tune the MCL model using the labeled training set of the target corpus. For example, to perform the domain adaptation with the Davidson dataset as the target corpus, we train a multi-corpus learning model with the MCL setup using Founta, Hateval, Waseem, and Wikipedia

training sets. We then take the trained MCL model and add a new set of corpus-specific layers. We then fine-tune the shared layers and the newly added corpus-specific layer with the training set of the Davidson dataset.
Additionally, we simulate low-resource scenarios for domain adaptation. We perform this by training the multi-corpus learning model with the MCL setup by using the entire training sets of four corpora. Then, for the target corpus, we only use 100, 200, 500, and 1000 labeled training training samples for adaptation. The trained model is then evaluated on the test set to report the model performance. The average macro-F1 results obtained on five target corpora in low-resource scenarios are presented in Table 6.4. The results of domain adaptation are compared to low-resource SCL, where the SCL model is fine-tuned with the varying amount of training data of the target corpus (same model as in section 6.5.1.4). The results obtained for domain adaptation for Davidson, Founta, Hateval, Waseem, and Wikipedia dataset as target datasets are presented in Figure 6.8a, 6.8b, 6.8c, 6.8d, and 6.8e respectively.

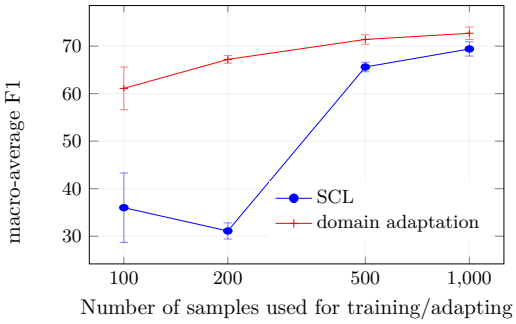
Approaches	Low-resource scenarios				All training samples for adaptation
	Number of adaptation samples				
	100	200	500	1000	
SCL (without adaptation)	49.4	53.4	67.7	70.2	74.4
MCL domain adaptation	**67.7**	**70.2**	**71.2**	**72.1**	**74.7**

Table 6.4: Average of macro-F1 for five test datasets for domain adaptation: low-resource scenario and all training samples.

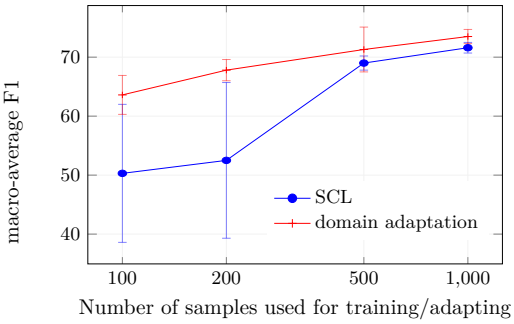
Table 6.4 shows that MCL domain adaptation performs better than SCL approach. We obtain a significant relative improvement of 37% (49.4 to 67.7) and 31.5% (53.4 to 70.2), using 100 and 200 training samples for adaptation, respectively. Figure 6.8 confirms these results for each target corpus. The improvement is higher when the amount of available training data is very low. This shows that the shared layers of multi-corpus model captures information from multiple related corpora, that can be transferred to a target corpus.
In the case of domain adaptation for the Wikipedia dataset, although the MCL setup model was trained with Davidson, Founta, Hateval, and Waseem datasets, where the samples were collected from Twitter, it showed a considerable amount of improvements in the low-resource domain adaptation. This shows that the MCL approach can still be helpful when the corpus is related but comes from a different domain.
Using the entire training set as target corpus for domain adaptation, an average macro-F1 of 74.7 is obtained. This result is slightly better than macro-F1 of 74.4 obtained using SCL.

6.6 Conclusion

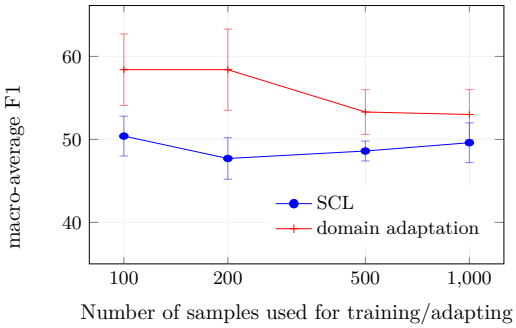
In this chapter, we explored multi-corpus learning (MCL) for hate speech classification. Our approach is based on the paradigm of multi-task learning, where one task corre-



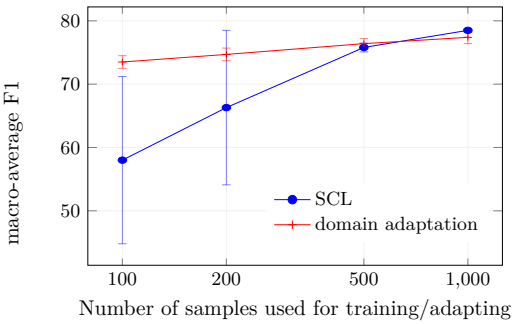
(a) Macro-averaged F1 on the Davidson test set. The MCL model is trained using Founta, Hateval, Waseem, and Wikipedia training sets. Model is adapted using varying amount of Davidson training set.



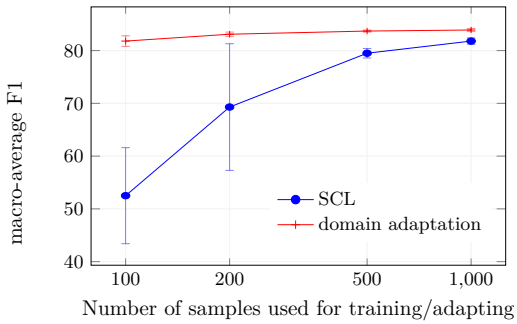
(b) Macro-averaged F1 on the Founta test set. The MCL model is trained using Davidson, Hateval, Waseem, and Wikipedia training sets. Model is adapted using varying amount of Founta training sets.



(c) Macro-averaged F1 on the Hateval test set. The MCL model is trained using Davidson, Founta, Waseem, and Wikipedia training sets. Model is adapted using varying amount of Hateval training sets.



(d) Macro-averaged F1 on the Waseem test set. The MCL model is trained using Davidson, Founta, Waseem, and Wikipedia training sets. Model is adapted using varying amount of Waseem training sets.



(e) Macro-averaged F1 on the Wikipedia test set. The MCL model is trained using Davidson, Founta, Hateval, and Waseem training sets. Model is adapted using varying amount of Wikipedia training sets.

Figure 6.8: Macro-average F1 for domain adaptation on Davidson, Founta, Hateval, Waseem and Wikipedia datasets.

sponds to one corpus. We used the pre-trained BERT model as shared layers to learn common representation for several corpora, and corpus specific layers to take into account corpus specific characteristics. We compared MCL with single-corpus learning (SCL) and showed that the MCL gives similar performance as SCL. Additionally, we showed that corpus-specific fine-tuning of MCL model yields better results than SCL and MCL. These results indicated that the MCL benefits from jointly training using multiple corpora and improves the classification performance when the model is fine-tuned on an individual corpus.

Furthermore, we simulated low-resource scenarios by reducing the amount of available training data. We showed that the MCL model captures the information from different tasks, and showed improved performance compared to SCL when the amount of available training data is low. Moreover, when the MCL model is further fine-tuned for a specific corpus, it shows significant performance gain compared to SCL. Our experiments were validated by downsampling the amount of available training data of every corpus by up to 100 samples. The results indicated that multi-corpus learning is beneficial when the amount of training samples is very low.

We combined the training sets of similar corpora into a single one. This increased the number of samples available to train the corpus-specific layers and reduced the number of corpus-specific layers. Our experiments showed that the SCL approach by combining corpora showed improvements when the class label definition used for the combined datasets were similar. The corpus-specific fine-tuning of MCL model showed improved results when compared with the SCL approach.

We used the MCL approach to perform supervised domain adaptation. We showed that a MCL model adapts to a new hate speech corpus better than the pre-trained BERT model, especially when the amount of available training data is very low. This showed that the shared layers representation learned by the MCL is helpful for domain adaptation. Overall, we experimentally showed the efficiency of MCL for low-resource hate speech classification and domain adaptation.

7 Conclusion and future research directions

This thesis was conducted within the framework of "**M**igration and **P**atterns of **Ha**te **S**peech **i**n **S**ocial Media - A Cross-Cultural Perspective" (M-PHASIS) project. We mainly aimed at deep learning-based classification approaches for hate speech classification and explored various solutions to improve the classification performance by increasing the number of training samples. In Section 7.1, we provide the conclusions and the limitations of our research work. We then discuss the perspectives and directions for future research in Section 7.2.

7.1 Summary

The Deep Neural Network (DNN) based classifiers were shown to be powerful for hate speech classification and they outperformed the classical machine learning-based classifiers. However, the performance of DNN-based classifier depend on the amount of available labeled training data. Thus, the small number of training samples in hate speech datasets is one of the dominant problems for classification models. This thesis addressed this issue by expanding the training set using three different approaches. Chapter 4, Chapter 5, and Chapter 6 addressed this issue using 'semi-supervised learning', 'data augmentation', and 'multi-corpus learning' approaches, respectively. Furthermore, we also evaluated these approaches in low-resource scenarios. We used macro-averaged F1 to evaluate the classification performance.

In **Chapter 4**, we combined large unlabeled data with the labeled data using semi-supervised learning to increase the number of training samples. Our semi-supervised learning was based on the label propagation algorithm.

- We showed that label propagation using pre-trained sentence representation (USE embeddings) gave poor performance. Through the analysis using PCA plots, we showed that in the pre-trained representations, samples belonging to different classes were not well separated.
- We proposed an approach using a multi-layer perceptron trained on a small amount of labeled data to transform pre-trained representations to task-specific representations. The PCA plots showed that in the task-specific representations samples from different classes were better separated.
- Label propagation using task-specific representations showed better results than

the baseline approach, especially when the amount of labeled data was low (few hundred samples). Thus, transforming pre-trained representations to task-specific representations was helpful for label propagation algorithm.
- We also showed that label propagation helps to improve the classifier performance in low-resource scenarios even when the unlabeled data comes from a different distribution.
- Overall, semi-supervised learning using label propagation helped to improve the classification results in low-resource scenarios.

The label propagation algorithm works by creating a huge $m \times m$, the size of this transition matrix increases with the number of samples in the labeled and the unlabeled set. This increases the memory and computational overhead.

In **Chapter 5**, we explored the data augmentation approach to increase the number of training samples. We generated synthetic samples, from the Generative Pre-trained Transformer-2 (GPT-2) model fine-tuned on conditional language model objective. The generated data were filtered using the BERT model to keep the most relevant samples. The filtered data were added to the original training data to train a hate speech classifier.
- Compared to the existing approach of generating data using several GPT-2 models, our approach of using a single GPT-2 model achieved higher classification scores.
- The filtering step, performed by the BERT, was fruitful. This was also shown using the qualitative evaluation using the Principal Component Analysis (PCA) plots.
- We showed that with an increase in the amount of augmented data, the performance of the classifier increases to a certain point and then saturates.
- Using our augmentation approach, we achieved a relative improvements of 3.5% (73.5 to 76.1) and 9.8% (68.7 to 75.4) on the Founta and Davidson datasets respectively.
- In low-resource scenarios, our approach provided significant performance improvements with few thousands training samples.

For our approach, we used C-GRU as the hate speech classifier, we need to evaluate the performance of other classifiers. Furthermore, our approach of filtering depends on the availability of a very good classifier to properly filtering the data. These limitations could be addressed in the future.

In **Chapter 6**, we explored multi-corpus learning to combine several available hate speech datasets to improve the classification performance. Our multi-corpus learning is based on the paradigm of multi-task learning. We used the pre-trained BERT model as the shared layers to harness the advantages of a pre-trained model. We used dense layers as task-specific layers to perform classification.
- We observed small improvements by corpus-specific fine-tuning of the multi-corpus model.
- In low-resource scenarios, our multi-corpus learning approach showed significant

improvements compared to the single-corpus learning approach.
- In low-resource scenarios, an additional step of fine-tuning the trained multi-corpus learning model showed an improvement. This showed that the multi-task learning approach is more helpful when the number of training samples is very low.

We performed supervised domain adaptation using the multi-corpus learning approach.
- The domain adaptation approach showed improvements compared to the single-corpus learning approach, especially in low-resource scenarios. This showed that the representations captured by multi-task learning from several hate speech corpora help to adapt to a new corpus.

The multi-corpus learning approach did not give significant performance improvements over single-corpus learning when trained on the entire training set. This problem could be addressed in the future.

7.2 Perspectives

7.2.1 Semi-supervised learning

Label probabilities as a measure of confidence: The label propagation algorithm assigns the probability that a data point belongs to a particular class. For our approach, we assigned the class label that has maximum probability as the label for the unlabeled data. However, the probability values can be used as a measure of confidence for the assigned label. The label confidence values can be used to perform per-sample-based scaling of loss (Ibrahim et al., 2020), such that misclassification of samples with higher confidence values can have a higher impact during training.

7.2.2 Data augmentation

Diverse data generation: The generated data can provide useful information to the classifier when these generated data are diverse. In our approach, the additional information in the generated data comes from the pre-training step, where the language model (GPT-2 in our case) was pre-trained with a huge amount of unlabeled data. One way to generate more diverse data is GPT-2 model can be continued to be pre-trained on more diverse hateful data.
The way of generating the data (sampling strategy) plays an important role in diversity of generated data. Diverse decoding schemes (Ippolito et al., 2019) can be explored in the future.

Alternate filtering strategy: The performance of the final classifier depends on filtering strategy. In our filtering method, the quality of filtering depends on the availability of good classifier. Furthermore, the top samples sorted by BERT come from a small region around the centroids of the target classes, indicating a lack of diversity in the filtered samples. Thus, we can explore an alternate filtering strategy to select relevant as well as a diverse set of samples for data augmentation.

Adversarial learning for data augmentation: Our data augmentation approach has three independent models: a model to generate the data, a model to filter the data, and a model to perform training on augmented data. All three models work independently. However, using adversarial networks such as Sequence Generative Adversarial Networks (SeqGAN) (Yu et al., 2017) or other adversarial training mechanisms, the generator and the discriminator (classifier) can be trained jointly.

Ensemble approaches: Various data augmentation techniques have been explored in the literature. Different data augmentation techniques come with different advantages. Combining several data augmentation techniques to benefit the advantage of all these techniques can be explored in the future.

7.2.3 Multi-corpus learning

Task (corpus) grouping and architecture search: Fifty et al. (2021) showed that grouping a few tasks based on inter-task affinity and training a model with a group of closely related tasks can help to improve the performance of multi-task learning approaches. Lu et al. (2017) suggested beginning with a thin model architecture, that dynamically splits into a tree-like structure during training, where similar tasks share the same branch of the tree. Such neural architecture search techniques to improve multi-task learning is little explored in the domain of Natural Language Processing (NLP) and hate speech classification and can be addressed in the future.

7.2.4 Long-term directions

Context as features: For the Twitter datasets, some comments are associated with URLs containing images or the parent comment. Such context information can be useful and used as additional input features for the classifier. Very few datasets in hate speech provide context information, hence, it is necessary to design such datasets. Additionally, very few works in hate speech classification have considered the context as additional features, future works can be conducted in the direction to understand the influence of context for hate speech classification tasks.

Annotator confidence and annotator noise: A few datasets such as Founta or Davidson provide additional details on annotations such as the number of annotators annotating the Tweet for a particular label. These features can be used to as additional features to the classifier. The annotation confidence can also be used to scale the loss during the training.

Explainability and in depth analysis: The DNN based classification decisions are often considered as black-box decisions, because the results of these models are difficult to interpret. However, with the advances in the explainability in the field of NLP (Danilevsky et al., 2020), these model's decisions can be interpreted. This enables us to better understand the classification errors made by the models.

8 Résumé étendu

L'augmentation considérable de l'utilisation d'Internet a encouragé les gens à exprimer leurs opinions par le biais de forums d'information, de médias sociaux, etc. Bien que l'internet présente de nombreux avantages, certaines personnes en font un usage abusif en diffusant de fausses informations, des discours de haine, etc.
Les discours de haine sont des comportements de communication antisociaux. Ils ciblent les minorités de la société en fonction de leur sexe, de leur religion, de leur origine ethnique, etc. Les discours de haine peuvent conduire à l'humiliation, à la menace ou à la violence à l'égard d'un individu ou d'un groupe (Delgado and Stefancic, 2014). Avec l'avènement d'Internet, les plateformes de médias sociaux et autres forums en ligne sont devenus des lieux communs pour la propagation des discours de haine. Les discours de haine en ligne sont interdits par la loi dans de nombreux pays. Les médias sociaux et plateformes en ligne doivent supprimer les contenus haineux publiés par leurs utilisateurs. Cependant, en raison du de l'énorme quantité de données publiées journellement, la modération manuelle du contenu en ligne prend beaucoup de temps et est très coûteuse. Il faut donc mettre en place des systèmes de détection automatique des discours haineux, qui peuvent être construits à l'aide de techniques d'apprentissage automatique et de traitement du langage naturel (TAL).

8.1 Motivation

Les systèmes classiques de classification des discours haineux utilisaient des caractéristiques lexicales en entrée des modèles. Diverses caractéristiques et classifieurs ont été explorés pour la tâche de classification des discours de haine. Cependant, avec l'évolution de l'apprentissage profond, Badjatiya et al. (2017) a montré que les classifieurs basés sur l'apprentissage profond tels que les réseaux neuronaux convolutifs (CNN) et la mémoire à long terme (LSTM) surpassaient les classificateurs traditionnels basés sur l'apprentissage automatique tels que les SVM et la régression logistique. Suite à cela, de nombreux chercheurs ont montré que les techniques d'apprentissage profond fonctionnent mieux que les techniques traditionnelles d'apprentissage automatique pour la classification des discours de haine.
En raison des progrès récents des techniques d'apprentissage profond, nous nous sommes concentrés, tout au long de cette thèse, sur la classification des discours haineux à l'aide de classifieurs basés sur des réseaux neuronaux profonds (DNN). La performance des DNN dépend de la quantité de données étiquetées disponibles pour les entraîner. Cependant, en raison du temps et du coût nécessaires à l'annotation, de nombreux corpus de discours haineux contiennent peu d'échantillons étiquetés. Afin d'augmenter

le nombre d'échantillons étiquetés pour entrainer un classifieur performant, nous explorons *l'apprentissage semi-supervisé*, *l'augmentation des données*, et *l'apprentissage multi-tâches*.

8.2 Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une technique permettant de combiner une petite quantité de données étiquetées avec une grande quantité de données non étiquetées pendant l'apprentissage (Abney, 2007) afin d'améliorer les performances des classifieurs.
Nous avons exploré l'apprentissage semi-supervisé basé sur la propagation d'étiquettes (Xiaojin and Zoubin, 2002) pour la tâche de classification des discours haineux. La propagation d'étiquettes est une technique de semi-supervision basée sur les graphes, analogue à l'algorithme des k-plus proches voisins. Elle suppose que les données proches les uns des autres dans l'espace de représentation ont tendance à avoir une la même étiquette. Cet algorithme s'appuie sur un graphe de distances qui capture la proximité des échantillons. Notre travail vise à montrer que ces représentations de phrases pré-entraînées (USE embeddings (Cer et al., 2018)) sont des représentations génériques et ne sont pas spécifiques à la tâche de détection de la haine. Ainsi, elles donnent des performances médiocres lorsqu'elles sont utilisées pour l'algorithme de propagation d'étiquettes. Par conséquent, nous proposons de transformer ces représentations génériques en représentations spécifiques à la tâche qui peuvent être plus utiles.

8.2.1 Approche proposée

Notre approche consiste à transformer les représentations pré-entraînées (générées par USE) en représentations spécifiques à la tâche. Pour cela, nous entraînons d'abord un classifieur (Multilayer perceptron) en utilisant le corpus étiqueté. Nous utilisons les embeddings USE comme entrée de ce modèle. Le vecteur obtenu au niveau de la couche cachéeobtenue de la couche cachée est utilisé comme la représentation spécifique à la tâche.
Après l'entraînement du classifieur MLP, nous générons les représentations spécifiques pour les données étiquetées et non étiquetées pour obtenir les représentations spécifiques à la tâche. Ces représentations spécifiques à la tâche sont utilisées comme entrée de l'algorithme de propagation d'étiquettes. L'algorithme de propagation d'étiquettes fournit des étiquettes pour les échantillons de l'ensemble non étiqueté. Ainsi, nous avons maintenant les étiquettes pour toutes les données d'apprentissage.
Enfin, nous combinons l'ensemble étiqueté et l'ensemble non étiqueté pour entraîner le classifieurs pour détecter des discours de haine.

8.2.2 Résultats et conclusion

Nous avons montré que la propagation d'étiquettes à l'aide des embeddings de phrases pré-entraînés donne de moins bonnes performances de classification que l'entraînement

du classifieur avec les seules données étiquetées. Nous avons utilisé un perceptron multicouche (MLP) pour transformer ces représentations pré-entraînées en représentations spécifiques à la tâche. Nos résultats montrent que l'utilisation de représentations spécifiques à la tâche facilite l'apprentissage semi-supervisé, en particulier dans les scénarios à faibles ressources. Nous avons également validé la distance intra-classe et inter-classe à l'aide de graphiques PCA.
Enfin, nous avons utilisé les échantillons non étiquetés de Twitter, dont la distribution est différente de celle des ensembles de données considérés. Les expériences ont révélé une amélioration des performances de classification dans les scénarios à faibles ressources.

8.3 Augmentation des données

Les techniques d'augmentation des données élargissent l'ensemble d'apprentissage en créant de nouveaux échantillons synthétiques à partir des données existantes. Ces échantillons synthétiques ajoutent de nouvelles informations aux données d'apprentissage afin d'améliorer les performances du modèle (Shorten and Khoshgoftaar, 2019).
Les techniques d'augmentation des données en TAL sont : (a) la réplication d'échantillons en effectuant des perturbations mineures dans le texte, telles que l'ajout ou la suppression d'un mot, l'échange de mots dans une phrase, le remplacement de mots par des synonymes, etc. (b) la transformation du texte à l'aide de la rétro-traduction : (Sennrich et al., 2016; Shleifer, 2019). (c) la génération du texte synthétique en utilisant des modèles d'auto-encodeurs. Nous explorons la génération de données à l'aide d'un modèle de langage GPT-2.

8.3.1 Approche proposée

Nous nous appuyons sur l'approche de Wullach et al. (2020). Cette méthode consiste à ajuster (fine-tuning) deux modèles GPT-2, l'un pour la classe " discours haineux " et l'autre pour la classe " discours non haineux ". Nous générons 600K échantillons pour chaque classe. De plus, nous filtrons les échantillons générés en utilisant un modèle BERT ajusté pour ne garder que les 100K meilleurs échantillons. Les échantillons filtrés sont ensuite utilisés pour augmenter l'ensemble d'entraînement original et entraîner un modèle de classification C-GRU pour la tâche de classification. Par rapport à l'approche de Wullach et al. (2020), notre approche de l'augmentation des données par génération de langage présente deux différences essentielles :

1. Au lieu d'une simple classification binaire en 'discours haineux' ou 'discours non haineux', nous effectuons une classification en trois classes: 'discours haineux', 'langage abusif', et 'discours normal'.
2. Inspiré par (Keskar et al., 2019), nous affinons un modèle de langage GPT-2 conditionné par la classe, par opposition à l'affinement spécifique à une classe de plusieurs modèles GPT-2 de Wullach et al. (2020).

Nous analysons également comment les performances de classification varient en fonction de la quantité d'échantillons générés, et étudions comment le filtrage des échantillons

générés affecte les performances de classification.

8.3.2 Résultats et conclusion

Nos expériences ont montré que l'ajout de quelques milliers d'échantillons générés par le modèle GPT-2 à l'ensemble d'entraînement permet un gain de performance significatif par rapport à l'approche de base consistant à entraîner le classificateur uniquement avec les données d'entraînement originales. De plus, nous avons exploré l'ajout d'échantillons uniquement à la classe 'haineuse', et avons montré des améliorations significatives dans la performance de classification.
Nous avons analysé la qualité des données générées en évaluant des classifieurs entraînés uniquement sur les données générées. Les résultats ont montré qu'un modèle entraîné avec une quantité suffisante de données générées peut dépasser les performances du modèle entraîné en utilisant uniquement les données d'entraînement originales. Nos résultats ont montré que l'utilisation du filtrage basé sur BERT aide à choisir des échantillons pertinents pour l'augmentation des données. Enfin, dans une configuration à faibles ressources, nous avons montré que notre approche d'augmentation des données permet un gain de performance même lorsque nous avons très peu d'échantillons d'entraînement étiquetés.

8.4 Apprentissage multi-tâches

Nous adaptons le paradigme de l'apprentissage multi-tâches à l'apprentissage multi-corpus pour la classification des discours de haine. L'apprentissage multi-tâches est motivé par la capacité des humains à apprendre simultanément plusieurs tâches connexes. La stratégie d'échantillonnage, la période pendant laquelle les commentaires sont recueillis et la définition des étiquettes de classe ajoutent à la diversité entre les divers corpus de données. La diversité des corpus disponibles rend un modèle appris sur un corpus inefficace pour tester sur un autre corpus, ce qui réduit la généralisation. La différence entre ces jeux de données empêche également de les combiner directement. Bien que ces ensembles de données diffèrent les uns des autres, ils ont un point commun : ils contiennent des données sur les discours de haine. En utilisant l'approche d'apprentissage multi-corpus, un modèle unique peut être entraîné conjointement sur plusieurs corpus de discours de haine améliorant ainsi les représentations partagées apprises par le modèle.

8.5 Approche proposée

Notre approche d'apprentissage multi-corpus est basée sur les travaux de Liu et al. (2019b). Les auteurs ont combiné une gamme de tâches NLP en utilisant les couches partagées représentées par le modèle BERT pré-entraîné et plusieurs groupes de couches spécifiques aux tâches. De même, nous utilisons le modèle BERT pré-entraîné comme couches cachées partagées pour l'architecture d'apprentissage multi-tâches. Nous adaptons le paradigme de l'apprentissage multi-tâches à l'apprentissage multi-corpus. Dans

notre approche, nous traitons chaque corpus comme une tâche de classification, donc une tâche correspond à un corpus. Notre approche de l'apprentissage multi-corpus et de l'adaptation au domaine est présentée sur la figure 8.1.

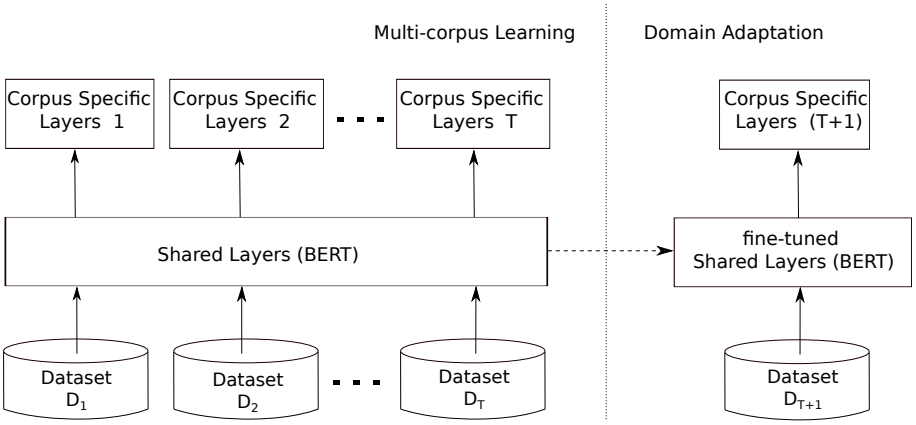


Figure 8.1: Architecture du système multicorpus (à gauche) et adaptation au domaine (à droite).

8.5.1 Résultats et conclusion

Nous avons montré que le réglage fin du modèle d'apprentissage multicorpus donne de meilleurs résultats que l'apprentissage monocorpus et l'apprentissage multicorpus. Ces résultats indiquent que le modèle d'apprentissage multi-corpus bénéficie de l'apprentissage conjoint de plusieurs corpus et améliore les performances de classification.
De plus, en utilisant une configuration à faibles ressources, nous avons montré que le modèle d'apprentissage multi-corpus capture l'information de différentes tâches, et obtient une meilleure performance par rapport à l'apprentissage mono-corpus lorsque la quantité de données d'entraînement disponible est faible.
Nous avons montré qu'un modèle d'apprentissage multi-corpus entraîné pour des tâches de classification de discours de haine s'adapte mieux à un nouveau corpus que le modèle BERT, en particulier lorsque la quantité de données d'entraînement disponibles est très faible. Ainsi, le modèle multi-corpus peut être utilisé efficacement pour la tâche d'adaptation au domaine.

Bibliography

Abburi, H., Parikh, P., Chhaya, N., and Varma, V. (2021). Fine-grained multi-label sexism classification using a semi-supervised multi-level neural approach. *Data Science and Engineering*, pages 1–21.

Abdollahi, M., Gao, X., Mei, Y., Ghosh, S., and Li, J. (2020). A dictionary-based oversampling approach to clinical document classification on small and imbalanced dataset. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 357–364.

Abney, S. (2007). *Semi-supervised learning for computational linguistics*. CRC Press.

Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). *arXiv preprint arXiv:1803.08375*.

Agrawal, S. and Awekar, A. (2018). Deep learning for detecting cyberbullying across multiple social media platforms. In *European conference on information retrieval*, pages 141–153.

Al Kuwatly, H., Wich, M., and Groh, G. (2020). Identifying and measuring annotator bias based on annotators' demographic characteristics. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 184–190.

Alatawi, H. S., Alhothali, A. M., and Moria, K. M. (2021). Detecting white supremacist hate speech using domain specific word embedding with deep learning and BERT. *IEEE Access*, 9:106363–106374.

Alexandrescu, A. and Kirchhoff, K. (2007). Data-driven graph construction for semi-supervised graph-based learning in NLP. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 204–211.

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., and Zwerdling, N. (2020). Do not have enough data? Deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

Arango, A., Pérez, J., and Poblete, B. (2020). Hate speech detection is not as easy as you may think: A closer look at model validation (extended version). *Information Systems*, page 101584.

Aroyehun, S. T. and Gelbukh, A. (2018). Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97.

Awasthi, A., Ghosh, S., Goyal, R., and Sarawagi, S. (2019). Learning from rules generalizing labeled exemplars. In *International Conference on Learning Representations*.

Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. (2017). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Banerjee, S., Chakravarthi, B. R., and McCrae, J. P. (2020). Comparison of pretrained embeddings to identify hate speech in indian code-mixed text. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 21–25.

Banerjee, S., Sarkar, M., Agrawal, N., Saha, P., and Das, M. (2021). Exploring transformer based models to identify hate speech and offensive content in english and indo-aryan languages. *arXiv preprint arXiv:2111.13974*.

Bansal, A., Kaushik, A., and Modi, A. (2021). IITK@ detox at semeval-2021 task 5: Semi-supervised learning and dice loss for toxic spans detection. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 211–219.

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel Pardo, F. M., Rosso, P., and Sanguinetti, M. (2019). SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63.

Beddiar, D. R., Jahan, M. S., and Oussalah, M. (2021). Data expansion using back translation and paraphrasing for hate speech detection. *Online Social Networks and Media*, 24:100153.

Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7:2399–2434.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Benk, M. (2019). *Data Augmentation in Deep Learning for Hate Speech Detection in Lower Resource Settings*. PhD thesis, Universität Zürich.

Blodgett, S. L., Green, L., and O'Connor, B. (2016). Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Bodapati, S., Gella, S., Bhattacharjee, K., and Al-Onaizan, Y. (2019). Neural word decomposition models for abusive language detection. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 135–145.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bojkovskỳ, M. and Pikuliak, M. (2019). Stufiit at semeval-2019 task 5: Multilingual hate speech detection on Twitter with muse and elmo embeddings. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 464–468.

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. *Advances in neural information processing systems*, 29:4349–4357.

Bosco, C., Felice, D., Poletto, F., Sanguinetti, M., and Maurizio, T. (2018). Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9.

Bowman, S., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Canete, J., Chaperon, G., Fuentes, R., and Pérez, J. (2020). Spanish pre-trained BERT model and evaluation data. *Practical ML for Developing Countries Workshop(PML4DC) at International Conference on Learning Representations(ICLR)*, 2020.

Cao, R. and Lee, R. K.-W. (2020). Hategan: Adversarial generative-based data augmentation for hate speech detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6327–6338.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Caselli, T., Basile, V., Mitrović, J., Kartoziya, I., and Granitzer, M. (2020). I feel offended, don't be abusive! Implicit/Explicit messages in offensive and abusive language. In *Proceedings of the 12th language resources and evaluation conference*, pages 6193–6202.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6.

Chen, Y., Zhou, Y., Zhu, S., and Xu, H. (2012). Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Chopra, S., Sawhney, R., Mathur, P., and Shah, R. R. (2020). Hindi-English hate speech detection: Author profiling, debiasing, and practical perspectives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 386–393.

Chung, Y.-L., Kuzmenko, E., Tekiroglu, S. S., and Guerini, M. (2019). CONAN-COunter NArratives through Nichesourcing: A multilingual dataset of responses to fight online hate speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2819–2829.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Cortes, C. and Vapnik, V. (1995). Support vector machine. *Machine learning*, 20(3):273–297.

Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., and Sen, P. (2020). A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459.

Davidson, T., Bhattacharya, D., and Weber, I. (2019). Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International Association for the AAAI Conference on Web and Social Media*, volume 11, pages 512–515.

Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M., and Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95.

Delgado, R. and Stefancic, J. (2014). Hate speech in cyberspace. *Wake Forest Law Review*, 49:319.

Dembowski, J., Wiegand, M., and Klakow, D. (2019). Language independent named entity recognition using distant supervision. In *Human Language Technologies as a Challenge for Computer Science and Linguistics. Proceedings of the 8th Language & Technology Conference*, pages 68–72.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dixon, L., Li, J., Sorensen, J., Thain, N., and Vasserman, L. (2018). Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Dror, R., Baumer, G., Shlomov, S., and Reichart, R. (2018). The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, page 1383–1392.

D'Sa, A. G., Illina, I., and Fohr, D. (2020). BERT and fasttext embeddings for automatic detection of toxic speech. In *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*, pages 1–5.

D'Sa, A. G., Illina, I., Fohr, D., Klakow, D., and Ruiter, D. (2020). Label propagation-based semi-supervised learning for hate speech classification. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 54–59.

D'Sa, A. G., Illina, I., Fohr, D., Klakow, D., and Ruiter, D. (2021). Exploring conditional language model based data augmentation approaches for hate speech classification. In *Text, Speech, and Dialogue: 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6-9, 2021, Proceedings*, volume 12848, page 135.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

ElSherief, M., Ziems, C., Muchlinski, D., Anupindi, V., Seybolt, J., De Choudhury, M., and Yang, D. (2021). Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363.

Fauzi, M. A. and Yuniarti, A. (2018). Ensemble method for Indonesian Twitter hate speech detection. *Indonesian Journal of Electrical Engineering and Computer Science*, 11(1):294–299.

Fersini, E., Rosso, P., and Anzovino, M. (2018). Overview of the task on automatic misogyny identification at IberEval 2018. *IberEval@ SEPLN*, 2150:214–228.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34.

Florio, K., Basile, V., Polignano, M., Basile, P., and Patti, V. (2020). Time of your hate: The challenge of time in hate speech detection on social media. *Applied Sciences*, 10(12):4180.

Fortuna, P., Soler, J., and Wanner, L. (2020). Toxic, hateful, offensive or abusive? What are we really classifying? An empirical analysis of hate speech datasets. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, pages 6786–6794.

Fortuna, P., Soler-Company, J., and Wanner, L. (2021). How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets? *Information Processing & Management*, 58(3):102524.

Founta, A. M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., Vakali, A., Sirivianos, M., and Kourtellis, N. (2018). Large scale crowdsourcing and characterization of Twitter abusive behavior. In *Twelfth International AAAI Conference on Web and Social Media*, pages 491–500.

Gambäck, B. and Sikdar, U. K. (2017). Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.

Gao, L. and Huang, R. (2017). Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 260–266.

Gao, L., Kuppersmith, A., and Huang, R. (2017). Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782.

Ghosh, A. and Veale, T. (2016). Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 161–169.

Gitari, N. D., Zuping, Z., Damien, H., and Long, J. (2015). A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Gomez, R., Gibert, J., Gomez, L., and Karatzas, D. (2020). Exploring hate speech detection in multimodal publications. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1470–1478.

Gröndahl, T., Pajola, L., Juuti, M., Conti, M., and Asokan, N. (2018). All you need is "Love": Evading hate speech detection. In *Proceedings of the 11th ACM workshop on artificial intelligence and security*, pages 2–12.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

GuillermoCarbonell, B. M. and MichaelWojatzki, B. N. (2016). Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *Bochumer Linguistische Arbeitsberichte*, pages 6–12.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328.

Hedderich, M. A. and Klakow, D. (2018). Training a neural network in a low-resource setting on automatically annotated noisy data. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 12–18.

Henderson, M., Al-Rfou, R., Strope, B., Sung, Y.-H., Lukács, L., Guo, R., Kumar, S., Miklos, B., and Kurzweil, R. (2017). Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Hosseini, H., Kannan, S., Zhang, B., and Poovendran, R. (2017). Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.

Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*.

Hu, R., Dorris, W., Vishwamitra, N., Luo, F., and Costello, M. (2020). On the impact of word representation in hate speech and offensive language detection and explanation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 171–173.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. (2017). Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Huang, Z., Eidelman, V., and Harper, M. (2009). Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 213–216.

Ibrahim, K. M., Epure, E. V., Peeters, G., and Richard, G. (2020). Confidence-based weighted loss for multi-label classification with missing labels. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 291–295.

Indurthi, V., Syed, B., Shrivastava, M., Chakravartula, N., Gupta, M., and Varma, V. (2019). Fermi at semeval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 70–74.

Ippolito, D., Kriz, R., Sedoc, J., Kustikova, M., and Callison-Burch, C. (2019). Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691.

Jaitly, N. and Hinton, G. E. (2013). Vocal tract length perturbation (VTLP) improves speech recognition. In *Proceedings of ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117, page 21.

Jha, A. and Mamidi, R. (2017). When does a compliment become sexist? Analysis and classification of ambivalent sexism using Twitter data. In *Proceedings of the second workshop on NLP and computational social science*, pages 7–16.

Jungiewicz, M. and Smywiński-Pohl, A. (2019). Towards textual data augmentation for neural networks: Synonyms and maximum loss. *Computer Science*, 20(1):57–83.

Kapil, P. and Ekbal, A. (2020). A deep neural network based multi-task learning approach to hate speech detection. *Knowledge-Based Systems*, 210:106458.

Karl Pearson F.R.S. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Khatri, C., Hedayatnia, B., Goel, R., Venkatesh, A., Gabriel, R., and Mandal, A. (2018). Detecting offensive content in open-domain conversations using two stage semi-supervision. *32nd Conference on Neural Information Processing Systems (NeurIPS)*.

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., and Testuggine, D. (2020). The hateful memes challenge: Detecting hate speech in multimodal memes. *Advances in Neural Information Processing Systems*, 33:2611–2624.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. *Advances in neural information processing systems*, 28.

Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2015). Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Kobayashi, S. (2018). Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

Kumar, R., Ojha, A. K., Malmasi, S., and Zampieri, M. (2018a). Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.

Kumar, R., Reganti, A. N., Bhatia, A., and Maheshwari, T. (2018b). Aggression-annotated corpus of Hindi-English code-mixed data. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Kumar, V., Choudhary, A., and Cho, E. (2020). Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lee, Y., Yoon, S., and Jung, K. (2018). Comparative studies of detecting abusive language on Twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 101–106.

Leonardelli, E., Menini, S., and Tonelli, S. (2020). DH-FBK@ HaSpeeDe2: Italian hate speech detection via self-training and oversampling. *EVALITA Evaluation of NLP and Speech Tools for Italian-December 17th, 2020*, page 110.

Levy, O. and Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Li, Y., Xu, L., Tian, F., Jiang, L., Zhong, X., and Chen, E. (2015). Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Liu, H. and Singh, P. (2004). Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Liu, P., Li, W., and Zou, L. (2019a). 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*, pages 87–91.

Liu, R., Xu, G., and Vosoughi, S. (2020). Enhanced offensive language detection through data augmentation. *arXiv preprint arXiv:2012.02954*.

Liu, X., He, P., Chen, W., and Gao, J. (2019b). Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019c). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. (2017). Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5334–5343.

Luu, S., Nguyen, K., and Nguyen, N. (2020). Empirical study of text augmentation on social media text in vietnamese. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pages 462–470.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., and Frieder, O. (2019). Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152.

Magu, R., Joshi, K., and Luo, J. (2017). Detecting the hate code on social media. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Mandl, T., Modha, S., Majumder, P., Patel, D., Dave, M., Mandlia, C., and Patel, A. (2019). Overview of the HASOC track at FIRE 2019: Hate speech and offensive content identification in Indo-European languages. In *Proceedings of the 11th forum for information retrieval evaluation*, pages 14–17.

Mao, Y., Xi, M., Yu, H., and Wang, X. (2011). Semi-supervised logistic regression via manifold regularization. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 23–28.

Melton, J., Bagavathi, A., and Krishnan, S. (2020). DeL-haTE: A deep learning tunable ensemble for hate speech detection. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1015–1022.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 33–40.

Mikolov, T., Chen, K., Corrado, G. S., and Jeffrey, D. (2013a). Efficient estimation of word representations in vector space. In *ICLR (Workshop Poster)*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mishra, P., Del Tredici, M., Yannakoudakis, H., and Shutova, E. (2018a). Author profiling for abuse detection. In *Proceedings of the 27th international conference on computational linguistics*, pages 1088–1098.

Mishra, P., Yannakoudakis, H., and Shutova, E. (2018b). Neural character-based composition models for abuse detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 1–10.

Moh, M., Moh, T.-S., and Khieu, B. (2020). No" love" lost: Defending hate speech detection models against adversaries. In *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–6.

Mou, G., Ye, P., and Lee, K. (2020). SWE2: Subword enriched and significant word emphasized framework for hate speech detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1145–1154.

Mozafari, M., Farahbakhsh, R., and Crespi, N. (2019). A BERT-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*, pages 928–940.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. (2016). Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

Nockleby, J. T. (2000). *Encyclopedia of the American Constitution*. Macmillan, 2nd edition.

Oak, R. (2019). Poster: Adversarial examples for hate speech classifiers. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2621–2623.

Pamungkas, E. W. and Patti, V. (2019). Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. In *Proceedings of the 57th annual meeting of the association for computational linguistics: Student research workshop*, pages 363–370.

Park, J. H. and Fung, P. (2017). One-step and two-step classification for abusive language detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45.

Park, J. H., Shin, J., and Fung, P. (2018). Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804.

Paul, D., Singh, M., Hedderich, M. A., and Klakow, D. (2019). Handling noisy labels for robustly learning from self-training data for low-resource sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 29–34.

Pavlopoulos, J., Sorensen, J., Dixon, L., Thain, N., and Androutsopoulos, I. (2020). Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perea, M., Duñabeitia, J. A., and Carreiras, M. (2008). R34d1ng w0rd5 w1th numb3r5. *Journal of Experimental Psychology: Human Perception and Performance*, 34(1):237.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Plaza-Del-Arco, F. M., Molina-González, M. D., Ureña-López, L. A., and Martín-Valdivia, M. T. (2021). A multi-task learning approach to hate speech detection leveraging sentiment analysis. *IEEE Access*, 9:112478–112489.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36.

Prasad, N., Saha, S., and Bhattacharyya, P. (2021). A multimodal classification of noisy hate speech using character level embedding and attention. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Pratiwi, N. I., Budi, I., and Jiwanggi, M. A. (2019). Hate speech identification using the hate codes for Indonesian tweets. In *Proceedings of the 2019 2nd International Conference on Data Science and Information Technology*, pages 128–133.

Qian, J., Bethke, A., Liu, Y., Belding, E., and Wang, W. Y. (2019). A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4755–4764.

Qian, J., ElSherief, M., Belding, E., and Wang, W. Y. (2018). Leveraging intra-user and inter-user representation learning for automated hate speech detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 118–123.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rajamanickam, S., Mishra, P., Yannakoudakis, H., and Shutova, E. (2020). Joint modelling of emotion and abusive language detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4270–4279.

Ramponi, A. and Plank, B. (2020). Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855.

Rani, P., Suryawanshi, S., Goswami, K., Chakravarthi, B. R., Fransen, T., and McCrae, J. P. (2020). A comparative study of different state-of-the-art hate speech detection methods in Hindi-English code-mixed data. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 42–48.

Rathpisey, H. and Adji, T. B. (2019). Handling imbalance issue in hate speech classification using sampling-based methods. In *2019 5th International Conference on Science in Information Technology (ICSITech)*, pages 193–198.

Razavi, A. H., Inkpen, D., Uritsky, S., and Matwin, S. (2010). Offensive language detection using multi-level classification. In Farzindar, A. and Kešelj, V., editors, *Advances in Artificial Intelligence*, pages 16–27.

Razo, D. and Kübler, S. (2020). Investigating sampling bias in abusive language detection. In *Proceedings of the fourth workshop on online abuse and harms*, pages 70–78.

Rezvani, N., Beheshti, A., and Tabebordbar, A. (2020). Linking textual and contextual features for intelligent cyberbullying detection in social media. In *Proceedings of the 18th International Conference on Advances in Mobile Computing & Multimedia*, pages 3–10.

Rizos, G., Hemker, K., and Schuller, B. (2019). Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 991–1000.

Rosenthal, S., Atanasova, P., Karadzhov, G., Zampieri, M., and Nakov, P. (2021). Solid: A large-scale semi-supervised dataset for offensive language identification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 915–928.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *Proceedings of the Association for the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829.

Ruder, S. and Plank, B. (2018). Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054.

Sap, M., Card, D., Gabriel, S., Choi, Y., and Smith, N. A. (2019). The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678.

Sap, M., Swayamdipta, S., Vianna, L., Zhou, X., Choi, Y., and Smith, N. A. (2021). Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. *arXiv preprint arXiv:2111.07997*.

Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*, pages 1–10.

Sennrich, R., Haddow, B., and Birch, A. (2016). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Sharifirad, S., Jafarpour, B., and Matwin, S. (2018). Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 107–114.

Shleifer, S. (2019). Low resource text classification with ULMFit and backtranslation. *arXiv preprint arXiv:1903.09244*.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

Simard, P. Y., LeCun, Y. A., Denker, J. S., and Victorri, B. (1998). Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274.

Søgaard, A. (2010). Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

van Aken, B., Risch, J., Krestel, R., and Löser, A. (2018). Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42.

Vargas, F. A., de Góes, F. R., Carvalho, I., Benevenuto, F., and Pardo, T. A. S. (2021). Contextual lexicon-based approach for hate speech and offensive language detection. *arXiv preprint arXiv:2104.12265*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Vidgen, B., Harris, A., Nguyen, D., Tromble, R., Hale, S., and Margetts, H. (2019). Challenges and frontiers in abusive content detection. In *Proceedings of the third workshop on abusive language online*, pages 80–93.

Wang, B., Spencer, B., Ling, C. X., and Zhang, H. (2008). Semi-supervised self-training for sentence subjectivity classification. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 344–355.

Wang, K., Lu, D., Han, C., Long, S., and Poon, J. (2020). Detect all abuse! Toward universal abusive language detection models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6366–6376.

Wang, W. Y. and Yang, D. (2015). That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563.

Warner, W. and Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26.

Waseem, Z. (2016). Are you a racist or am i seeing things? Annotator influence on hate speech detection on Twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142.

Waseem, Z., Chung, W. H. K., Hovy, D., and Tetreault, J. (2017a). Proceedings of the first workshop on abusive language online. In *Proceedings of the First Workshop on Abusive Language Online*.

Waseem, Z., Davidson, T., Warmsley, D., and Weber, I. (2017b). Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84.

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Student Research Workshop*, pages 88–93.

Waseem, Z., Thorne, J., and Bingel, J. (2018). Bridging the gaps: Multi task learning for domain transfer of hate speech detection. In *Online harassment*, pages 29–55.

Wei, J. and Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389.

Wich, M., Al Kuwatly, H., and Groh, G. (2020). Investigating annotator bias with a graph-based approach. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 191–199.

Wich, M., Eder, T., Al Kuwatly, H., and Groh, G. (2021). Bias and comparison framework for abusive language datasets. *AI and Ethics*, pages 1–23.

Widmann, N. and Verberne, S. (2017). Graph-based semi-supervised learning for text classification. In *Proceedings of the ACM SIGIR international conference on theory of information retrieval*, pages 59–66.

Wiegand, M., Ruppenhofer, J., and Kleinbauer, T. (2019). Detection of abusive language: The problem of biased datasets. In *Proceedings of the 2019 conference of the North American Chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long and short papers)*, pages 602–608.

Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Worsham, J. and Kalita, J. (2020). Multi-task learning for natural language processing in the 2020s: Where are we going? *Pattern Recognition Letters*, 136:120–126.

Wu, X., Lv, S., Zang, L., Han, J., and Hu, S. (2019). Conditional BERT contextual augmentation. In *International Conference on Computational Science*, pages 84–95.

Wulczyn, E., Thain, N., and Dixon, L. (2017). Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

Wullach, T., Adler, A., and Minkov, E. (2020). Towards hate speech detection at large via deep generative modeling. *IEEE Internet Computing*, 25(2):48–57.

Xiaojin, Z. and Zoubin, G. (2002). Learning from labeled and unlabeled data with label propagation. *Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University.*

Xu, Z. and Zhu, S. (2010). Filtering offensive language in online communities using grammatical relations. In *Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pages 1–10.

Yang, J. and Leskovec, J. (2011). Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Yin, W. and Zubiaga, A. (2021). Towards generalisable hate speech detection: A review on obstacles and solutions. *PeerJ Computer Science*, 7:e598.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Yu, N., Pan, S., Yang, C.-c., and Tsai, J.-Y. (2020). Exploring the role of media sources on COVID-19–related discrimination experiences and concerns among asian people in the United States: Cross-sectional survey study. *Journal of Medical Internet Research*, 22(11):e21684.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019a). 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019b). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 28:649–657.

Zhang, Y. and Yang, Q. (2021). A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Zhang, Z., Robinson, D., and Tepper, J. (2018). Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In *European semantic web conference*, pages 745–760.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

Zhou, Y. and Goldman, S. (2004). Democratic co-learning. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602.

Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Zimmerman, S., Kruschwitz, U., and Fox, C. (2018). Improving hate speech detection with deep learning ensembles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.