Achieving Hate Speech Detection in a Low Resource Setting

Peiyu Li
Utah State University

ACHIEVING HATE SPEECH DETECTION IN A LOW RESOURCE SETTING

by

Peiyu Li

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Shuhan Yuan, Ph.D. Major Professor	Xiaojun Qi, Ph.D. Committee Member
Dean Mathias, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

Achieving Hate Speech Detection in a Low Resource Setting

by

Peiyu Li, Master of Science

Utah State University, 2021

Major Professor: Shuhan Yuan, Ph.D.
Department: Computer Science

With the rise of social networks, social media has become the major platform for people to share and exchange information. Due to the anonymous environments provided by social networks, some malicious users express the offensive content online which contributes to increasing hate speech cases. To prevent the negative effects of hate speech, automatically hate speech detection methods are required. Models based on machine learning and natural language processing provide a way to detect hate speech. However, annotating sufficient data to train these models is a big challenge. In this thesis, we focus on achieving hate speech detection based on limited labeled data. In particular, we propose three research tasks to address the low resource data problem. First, we propose a hate speech detection model based on fine-tuning a pre-trained language model BERT on limited annotated data. By transferring the knowledge from the pre-trained model into our low-resource hate speech, we expect to get decent classification performance by just using a small size of labeled training data. Second, we propose a multitask learning approach to conduct hate speech detection. By applying multitask learning for detecting hate speeches on different platforms, multiple hate speech classifiers are trained jointly. In this case, we can leverage the correlations among different hate speech detection tasks to improve the detection performance. Last, we propose a domain adversarial neural network for hate speech detection.

In this scenario, we train the classification model on data from some social network platforms (source domains) and apply the classifier trained on source domains to detect the hate speeches on a new platform (target domain). By using the domain adaptation technique, we make full use of the labeled data from the source domain and achieve the domain transfer to our target domain. Empirical studies show that our proposed approaches can achieve good performance on hate speech detection in a low resource setting.

(72 pages)

PUBLIC ABSTRACT

Achieving Hate Speech Detection in a Low Resource Setting

Peiyu Li

Online social networks provide people with convenient platforms to communicate and share life moments. However, because of the anonymous property of these social media platforms, the cases of online hate speeches are increasing. Hate speech is defined by the Cambridge Dictionary as “public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation”. Online hate speech has caused serious negative effects to legitimate users, including mental or emotional stress, reputational damage, and fear for one’s safety. To protect legitimate online users, automatically hate speech detection techniques are deployed on various social media. However, most of the existing hate speech detection models require a large amount of labeled data for training. In the thesis, we focus on achieving hate speech detection without using many labeled samples. In particular, we focus on three scenarios of hate speech detection and propose three corresponding approaches. (i) When we only have limited labeled data for one social media platform, we fine-tune a per-trained language model to conduct hate speech detection on the specific platform. (ii) When we have data from several social media platforms, each of which only has a small size of labeled data, we develop a multitask learning model to detect hate speech on several platforms in parallel. (iii) When we aim to conduct hate speech on a new social media platform, where we do not have any labeled data for this platform, we propose to use domain adaptation to transfer knowledge from some other related social media platforms to conduct hate speech detection on the new platform. Empirical studies show that our proposed approaches can achieve good performance on hate speech detection in a low resource setting.

To my family and friends....

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my major professor Dr. Shuhan Yuan. He is patient, professional, and always giving me motivational support during my Master's program. I am also thankful for the timely feedback and support from my supervisory committee members Dr. Xiaojun Qi and Dr. Dean Mathias. I would also like to thank Dr. Haitao Wang for giving me valuable advice in the first year of my Master's program. In addition, I am happy to have my friends and lab partners accompany me during the past two years: Haixuan, Meng, Anny, Kuan, Yiming, Xiankun, Jason, Princy, Namita, Knak, and Aashish. And huge thanks for my family's support and encouragement, which gives me a lot of energy and confidence. Finally, thanks to Kaitlyn, Genie, Cora, Vicki for their constant help with all kinds of questions.

Peiyu Li

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Online social media plays a very important role in our social life, which influences our lives in both positive and negative ways. It provides people platforms to communicate with each other online, where people can be informed about all kinds of news while just sitting at home. It is very convenient for people to share some interesting, fun and informative moments in public. However, there are many cases of public speeches called hate speeches that express hate or encourage violence, which cause varying degrees of harm to individuals or organizations.

According to Munro's study of the bad effects of hate speech on children, 30% of secondary school children in England have been deliberately targeted, threatened, or humiliated online. The hate speech causes very strong negative feelings, fear, and a sense of helplessness, which is even worse than offline bullying. These will further lead to school failure, depression, anxiety, and psychological problems [8]. Besides, the nationally representative Pew Research Center survey of 4,248 U.S. adults shows that about 41% of Americans have experienced personally harassing behavior online, and 66% of them have witnessed these behaviors directed at others; for those who experience online harassment directly, they might have profound real-world consequences, ranging from mental or emotional stress to reputational damage or even fear for one's safety [9]. Moreover, according to the 2015 hate crime statistic [10], hate crime cases have been increasing rapidly. Two main factors can explain this. On the one hand, on the internet, social networks provide anonymous environments, which make people more likely to conduct aggressive behavior [11]. On the other hand, the willingness to express emotional content online is increasing, which also contributes to the spread of hate speech [12].

To prevent the harmful effect on society from this kind of prejudiced communication, automatically hate speech detection and prevention tools are required on social network platforms. Automated techniques, which aims to classify hate speeches in a timely manner so that the online users can be protected from social hate [13, 14].

In general, hate speech detection application scenarios can be classified into three categories: (i) High- or Rich-resource settings, where we can get access to a large amount of annotated data; (ii) Low-resource or Resource-poor ones, where only limited annotated data are available; and (iii) Zero-resource settings, where there is no annotated data available in the target context. Most of the approaches used for hate speech detection tasks are supervised methods [15], which require manual labeling of a large volume of text. The labeling task is very laborious but it is proved to be very efficient for domain-independent events [15].

In this thesis, we adopt three annotated datasets that are from different social media platforms for hate speech detection. Firstly, we make full use of the three annotated datasets to apply traditional machine learning models and deep learning models on hate speech detection based on the scenario (i), our result shows good performance in this scenario. However, in many real-world scenarios, it is usually prohibitive or expensive to obtain large amounts of labeled data. Considering the scenario (ii), where only limited annotated data are available, we propose transfer learning and multitask learning(MTL) approaches to address the low resource problem. By using transfer learning, we can transfer the knowledge from the high resource domains to a new low-resource target domain. For many applications in NLP, most popular transfer learning methods choose to pre-train a large language model, e.g., ELMo [16], GPT [17], and BERT [6]. In this thesis, we choose Bidirectional Encoder Representations from Transformers (BERT) as the pre-trained language model. BERT is trained on a large number of words and articles from Wikipedia and the Book Corpus. By importing a pre-trained BERT and the pre-trained BERT model then can be fine-tuned with just one additional output layer to create a powerful hate speech classification neural network. For MTL, Caruana [18] summarizes the goal of MTL succinctly: "MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks". We propose a multitask learning model that achieves hate speech detection on three related tasks. Specifically, we train these three tasks in parallel while using a shared representation. Therefore, what is learned for each task can help other tasks be learned better. At last, for scenario (iii), where no annotated data in the target context is available, we propose domain adversarial neural networks to detect hate speech. Domain adversarial neural network is directly inspired by the theory on domain adaptation [19], which aims at adapting a classifier trained on the source domain for use in the target domain, and the performance in the target domain depends on both the performance in the source domain and the similarity between the source domain and target domain [19]. In this situation, we leverage the annotated data from labeled source domain to detect the hate speeches in the target domain without using labeled data.

1.1 Hate Speech Definition

Detecting hate speech is not an easy task, even for humans. In this section, we first describe several definitions of hate speech in literature. Hate speech is defined by the Cambridge Dictionary as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation" [20]. According to the Twitter platform, any contents that promote violence against or directly attack or threaten other people based on race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease are regarded as hate speech [21]. The YouTube community defines hate speech as contents that promotes violence or hatred against individuals or groups based on certain attributes, such as race or ethnic origin, religion, disability, gender, age, veteran status, and sexual orientation/gender identity [22]. The YouTube community also emphasizes that there is a fine line between what is and what is not considered to be hate speech; for instance, it is generally okay to criticize a nation-state, but not okay to post malicious hateful comments about a group of people solely based on their ethnicity [22]. According to The International Lesbian, Gay, Bisexual, Trans and Intersex Association (ILGBTIA, and in Europe ILGA), hate speech is public expressions that spread, incite, promote or justify hatred, discrimination, or hostility towards a specific group [23]. By comparing those definitions from different sources, we can conclude that hate speech aims at attacking or diminishing specific targets to incite violence or hate. Several Examples of hate speech are 1-3 [1].

1. At least I'm not a nigger.
2. California is full of white trash.
3. Why people think gay marriage is okay is beyond me. Sorry, I don't want my future son to see 2 fags walking down the street holding hands.

Besides hate speech, there are many other related concepts, like hate [25], cyberbullying [26], abusive language [27], discrimination [28], toxicity [29], flaming [30]. All of these concepts are slightly distinct from but still related to hate speech. By exploring those concepts can give insight into how to automatically detect hate speech.

Table 1.1: Types of hate speech and examples (Table from Silva et al [1])

Categories	**Example of possible targets**
Race	black people, white people
Behavior	insecure people, sensitive people
Physical	obese people, beautiful people
Sexual orientation	gay people, straight people
Class	ghetto people, rich people
Gender	pregnant people, cunt, sexist people
Ethnicity	chinese people, indian people, paki
Disability	retard, bipolar people
Religion	religious people, jewish people
Other	drunk people, shallow people

[1]The examples are to illustrate the severity of the hate speech problem. They are taken from the Twitter dataset [24] that was used in our experiments and in no way reflect the opinion of the authors

1.2 Types of Hate Speech

Based on [1], the hate speech can be grouped into ten categories: Race, Behavior, Physical, Sexual Orientation, Class, Gender, Ethnicity, Disability, Religion, and Others. Table 1.1 shows the categories and their corresponding examples of possible targets.

1.3 Methodologies for Hate Speech Detection

In general, hate speech detection is a text classification task. Following the typical procedure for the text classification, we first need to extract the features from text data and then apply the classification models to detect the hate speech.

1.3.1 Features Representation for Hate Speech Detection

To apply classification models to detect the hate speech, text features need to be generated. Here, several commonly used approaches are presented

Dictionaries and Lexicons. This feature is commonly employed in unsupervised machine learning scenarios [31]. Wiegand et al. [32] propose novel features employing information from both corpora and lexical resources to achieve profane word detection. They build the lexicon by using general-purpose lexical resources. Since dictionaries based approaches generally suffer from the inability to find opinion words with domain and context specific orientations [33], they are usually not competitive compared with other features used in supervised approaches.

Bag-of-Words (BOW) and N-grams. BOW creates a corpus based on the words that occur in the training data, then the occurrence of each word is regarded as a feature for training. It is easy to implement but the word sequence, syntactic and semantic content are ignored. In this case, the performance is not guaranteed if the words are used in different contexts. Then N-grams is adopted to deal with this limitation. N-grams representation means a sequence of N adjacent words. It aims at enumerating all the expressions of size N and counting all occurrences. Since it incorporates at some degree the context of each word, the classifiers' performance is improved [34]. Except using words, we can also use N-grams with characters or syllables. For the abusive language detection, character N-gram features are proved to be more predictive than token N-gram features [35].

TF-IDF. The term frequency-inverse document frequency (TF-IDF) is a measure of the importance of a word in a document within a corpus and increases in proportion to the number of times that a word appears in the document [34]. By using TF-IDF, we can remove some n-grams from features based on their occurrence frequency in our corpus. In general, the high-frequency n-grams (eg. stop words) might be ignored since they are not helpful for us to discriminate texts; and low-frequency n-grams (eg. typos, rare n-grams) might be ignored since they may cause overfit.

Word Embedding. Word embedding brings up an extra semantic feature by generating distributed representations that introduce dependence between words, which mitigates the data sparsity problem [15]. Word2Vec and FastText are two commonly used techniques to construct word embedding [36,37]. According to Lilleberg et al. [37], word2vec is compatible with both supervised and unsupervised machine learning models.

1.3.2 Traditional Machine Learning for Hate Speech Detection

After generating feature representation from the corpus, we can apply classification algorithms to perform the detection task. In general, machine learning approaches are categorized into: supervised, semi-supervised and unsupervised approaches.

Supervised learning relies on a large volume of labeled texts. Data labeling is usually time-consuming but it is very efficient for domain-specific tasks. Most of the approaches for hate speech detection tasks are supervised methods [15].

Unsupervised learning is capable of handling a diversity of content while maintaining scalability because it does not require the manually labeled data [38]. It not only saves human labor but also dynamically extracts domain-related key terms.

Semi-supervised learning uses both labeled and unlabeled data during the training process. Hua et al. [39] prove the effectiveness of semi-supervised models in targeted-interest event detection tasks. They argue that the ability of unsupervised learning to handle small-scale events is limited, while supervised learning can effectively capture small-scale events but the scalability of the model decreases due to the need to manually label the data set [39]. Then they come up with the semi-supervised model to balance between these two situations.

1.3.3 Deep Learning for Hate Speech Detection

Deep learning methods have achieved notable performance in many classification tasks [40]. Unlike traditional machine learning methods, deep learning methods can automatically learn latent representations of the input data to perform classification [41]. Deep learning approaches have been widely applied to various natural language processing tasks, including text classification [42, 43]. Many recent studies adopt deep learning methods to detect hate speech in social media [35,36,44–50]. Mehdad et al. [35] apply Recurrent Neural Network (RNN) for hate speech detection. Gambäck et al. [47] conduct hate speech detection using Convolutional Neural Network (CNN). Badjatiya et al. [36] combine Long-Short Term Memory (LSTM) model and Gradient-Boosted Decision Tree to detect hate speech on Twitter. Zhang et al. [50] propose a new neural network architecture that combines CNN with Gated Recurrent Unit (GRU) to classify hate speech.

1.4 Organization of the Thesis

The rest of this thesis is organized as follows:

- In Chapter 2, we firstly introduce the commonly used traditional machine learning models and deep learning models for hate speech detection on Twitter, Wikipedia, and White Supremacy Forum. We conduct experiments based on the rich resource setting, we make full use of the data from Twitter, Wikipedia, and White Supremacy Forum, then compare the performances of various hate speech detection models in terms of different evaluation metrics.

- In Chapter 3, we focus on achieving effective hate speech detection while we can only get access to limited annotated data. To address the insufficient labeled data issue, we use the pre-trained language model BERT for hate speech classification on Twitter, Wikipedia, and White Supremacy Forum three social media platforms based on the low-resource setting. For each social media platform, we train a classifier that fine-tunes a pre-trained BERT model to classify hate speech. In this case, the knowledge from a pre-trained model can be transferred into our low-resource hate speech languages. In terms of low-resource setting, we propose a pre-trained BERT model and fine-tuning it on our hate speech detection tasks, then compare their performances with the previous techniques introduced in Chapter 2.

- In Chapter 4, we propose the multitask learning (MTL) method to deal with the low-resource problem. Contrary to single-task learning, which learns one task at one time during the training process, the idea behind MTL is to improve classification by learning tasks parallel. Here, we apply MTL on Twitter, Wikipedia, and White Supremacy Forum for three hate speech detection tasks. Since the three hate speech classification tasks are learning jointly, we can leverage potential correlations among three hate speech detection tasks for different social media platforms to extract common features. In this case, the corpus size of our training data is increased implicitly. The effectiveness of multitask learning will be validated by comparing it with single-task learning where only limited training data is available.

- In Chapter 5, we propose the domain adversarial neural network (DANN) on Twitter, Wikipedia, and White Supremacy Forum in terms of zero-resource setting. DANN is directly inspired by the theory on domain adaptation, which leverages sufficient annotated data from the related source domain to the unlabeled target domain. In this scenario, we don't have annotated data from the target domain, but the labeled data from the source domain is available. Here, we combine three social media platform datasets (Twitter, Wikipedia, and White Supremacy Forum) to build the domain datasets. We take each of Twitter, Wikipedia, and White Supremacy Forum as target domains in turn, and the rest as source domains, then conduct hate speech detection on each source-target domain pair. By applying DANN, the mappings between the source and the target domains will be built, so the classifier learned for the source domain can be applied to the target domain.

CHAPTER 2

Machine Learning for Hate Speech Detection

How do we conduct hate speech detection while we have enough data?

2.1 Introduction

The massive increase of user-generated web content has contributed to a great amount of hate speech on the web and social media platforms. As social media platforms become mainstream means for people to acquire and exchange information, the raging hate speech will absolutely affect our normal life. Therefore, it is urgent to automatically detect and filter hate speech. Over the past few years, interest in hate speech detection has continuously grown. A number of studies that aim at detecting hate speech have been come up with recently. A diverse range of classic machine-learning strategies has been proposed [24, 51–53]. These machine learning strategies usually include an initial feature-extraction phase. For example, Bag-of-Words vectors or Term-Frequency Inverse-Document-Frequency scores will be generated firstly; then these features are regarded as input for machine learning methods such as Logistic Regression, SVM, or Naive Bayes. In recent years, deep learning methods have attracted more interest to hate speech detection [36, 47–50, 54]. Compared with traditional machine learning methods, deep learning methods can achieve feature extraction and classification at the same time. Deep learning architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Gated Recurrent Units (GRUs), are predominantly used methods for several Natural Language Processing tasks [42]. In the context of hate speech, effectiveness has been proved with the use of CNNs [47, 49], GRUs [50], LSTMs [36, 54].

In this chapter, we conduct our experiments with annotated datasets that are from three social network platforms, Twitter, Wikipedia, and White Supremacy Forum. The details of these three datasets are described in the following sections. We apply both deep learning neural network architectures and traditional machine learning methods that are predominantly used in the task of hate speech detection to classify hate speech. Then we evaluate their performances in terms of macro averaged precision, recall, and F1 score respectively.

2.2 Methodology

2.2.1 Traditional Machine Learning Methods

- **Naïve Bayes (NB).** NB classifiers are probabilistic classifiers based on Bayes theorem Algorithm, which applies Bayes' theorem with strong independence assumptions between the features. While conducting text classification, NB is a popular baseline method for classifying sentences into different categories with word frequencies as the feature. The basic idea of NB to find the probabilities of categories given a text by using the joint probabilities of words and categories. It is based on the assumption of word independence. The starting point is the Bayes' theorem for conditional probability, stating that, for a given data point x and class C:

  $$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \tag{2.1}$$

  Furthermore, by making the assumption that for a data point $x = \{x_1,x_2,...x_n\}$, the probability of each of its attributes occurring in a given class is independent. Then the probability of x being categorized as class C can be estimated as follows:

  $$P(C|x) = P(C)\prod_{i=1}^{n} P(x_i|C) \tag{2.2}$$

- **Linear Support Vector Machine (SVM).** SVM classifiers are supervised learning models. The goal of SVM classifiers is to find the optimal hyperplane separating two different classes of data that will generate the best model for future data. The SVM method is proposed by Vapnik [55], and the great performance in pattern recognition and text categorization has been validated in [56]. In linear classification, SVM creates a hyperplane that separates the data into two sets with the maximum-margin. A hyperplane with the maximum-margin has the distances from the hyperplane to points when the two sides are equal. Mathematically, SVM learns the sign function

$$f(x) = sign(wx + b), \tag{2.3}$$

where ω is a weighted vector in $\mathcal{R}$. SVM finds the hyperplane

$$y = wx + b \tag{2.4}$$

by separating the space $\mathcal{R}$ into two half-spaces with the maximum-margin.

- **Logistic Regression.** Logistic regression is an advanced linear regression technique used for classifying both linear and non-linear data. It is commonly used to model data with binary responses. Logistic regression is a machine learning technique that is implemented by taking the given input value and multiplying the input with weight value [57]. Consider a feature vector x of one input text and the category C of this input text, logistic regression directly estimates the vector (w) of weights, then we use logistic regression models the prediction of specific text is hate speech or not. Mathematically, we have

$$\begin{aligned} C &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + e^{-(w \cdot x + b)}}, \end{aligned} \tag{2.5}$$

where the text falls into one of two class categories, hate speech or non-hate speech. C indicates this text belongs to which category, which is given as

$$C = \begin{cases} 0, & non\_hate; \\ 1, & hate \end{cases} \tag{2.6}$$

2.2.2 Deep Learning Based Methods

- **Convolutional Neural Network (CNN).** CNN is a kind of deep, feed-forward artificial neural network. CNN is commonly used in computer vision, which shows promising performance while applying to NLP tasks as well. The model architecture, shown in Fig 2.1, is a shallow-and-wide CNN from Kim2014 [58].

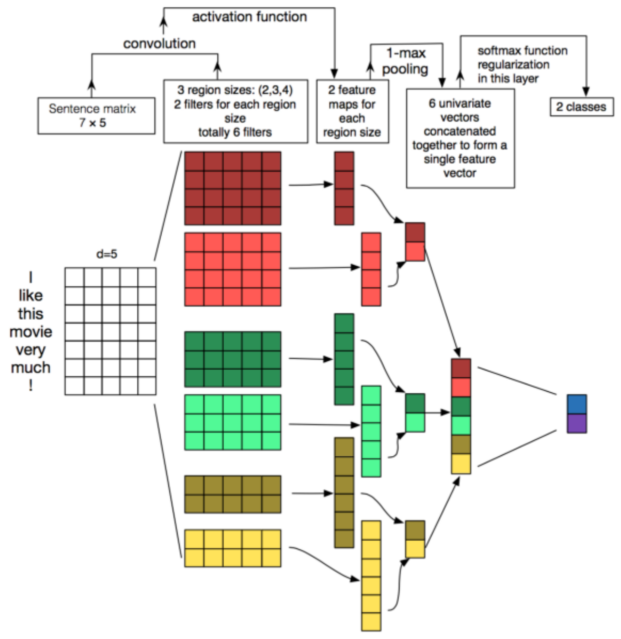

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Fig. 2.1: Architecture of the CNN model (Source: from [2])

We assume $x_i \in \mathcal{R}^d$ is the d-dimensional word vector corresponding to the i-th word in the sentence. Then we can denote a sentence of length n as

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n, \tag{2.7}$$

where $\oplus$ defines the concatenation operator. In this case, a $n \times d$ sentence matrix for a sentence of length n can be generated. Then we perform convolution on the sentence matrix via filters. A convolution operation involving a filter $W \in \mathcal{R}^{h \times d}$, which is applied to a window size of h to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by

$$c_i = f(W \cdot x_{i:i+h-1} + b), \tag{2.8}$$

where $b \in \mathcal{R}$ is a bias term and f is a nonlinear function such as the hyperbolic tangent. After applying this filter to each possible window of words in the sentence $\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$, we can get a feature map

$$c = [c_1, c_2, ..., c_{n-h+1}], \tag{2.9}$$

where $c \in \mathcal{R}^{n-h+1}$. Here, we use 2 filters for the same region size to learn complementary features from the same regions. We also use multiple filters with 2,3 and 4 region sizes. In this condition, the dimensionality of the feature map generated by each filter will vary as a function of the filter region size. A pooling function is thus applied to each feature map to induce a fixed-length vector. We use 1-max pooling which is to capture the most important feature (one with the highest value) [59]. For each feature map, it takes the maximum value $\hat{c} = max\{c\}$ as the feature corresponding to this particular filter. (In the following, we denote the feature extraction process as $h(x) = CNN(x)$.) After that, we concatenate the outputs generated from each filter map into a fixed-length, "top-level" feature vector. Then the feature vector is fed through a softmax function to generate the final classification.

- **Simple Recurrent Neural Network (Simple RNN).** A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. Fig 2.2 shows the architecture of RNN. RNN reads a sequence of tokens one by one. At each step, a recurrent network receives a new input vector (x_i) and the previous network state (h_{i-1}). A recurrent neuron stores the state of a previous input and combines with the current input, then a new state (h_{i+1}) is being computed. The new state contains the information about the current input and the previous states, thereby preserving some relationships between the current input token and the previous input tokens. After we get the final state output vector, then it can be passed to a fully connected layer for final prediction.

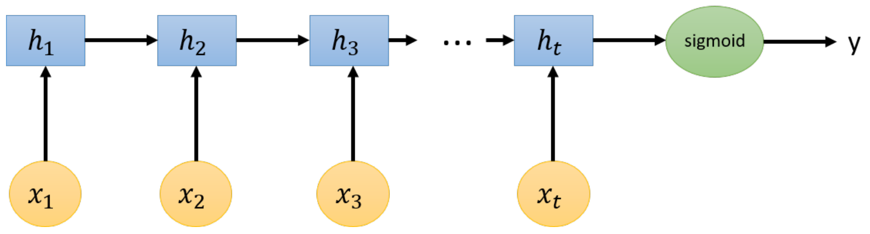


Fig. 2.2: Recurrent neural network

RNN models word sequence x as follows,

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.10}$$

$$y_t = \sigma_y(W_y h_t + b_y), \tag{2.11}$$

where x_t is input vector, h_t is hidden layer vector, y_t is output vector; W, U and b are parameter matrices and vector; σ_h and σ_y are activation functions. Equation 2.10 shows the connection between the previous and the current hidden states, which makes the information of previous context can be absorbed.

- **Long Short Term Memory (LSTM).** RNN suffers from vanishing and exploding gradients problems when the error of the gradient descent algorithm is back-propagated through the network [60], which makes RNN can not remember all input history effectively. In order to deal with these problems, Hochreiter et al. [61] propose LSTM that preserves long-term dependencies in a more effective way. Fig 2.3 shows the structure of the LSTM cell.

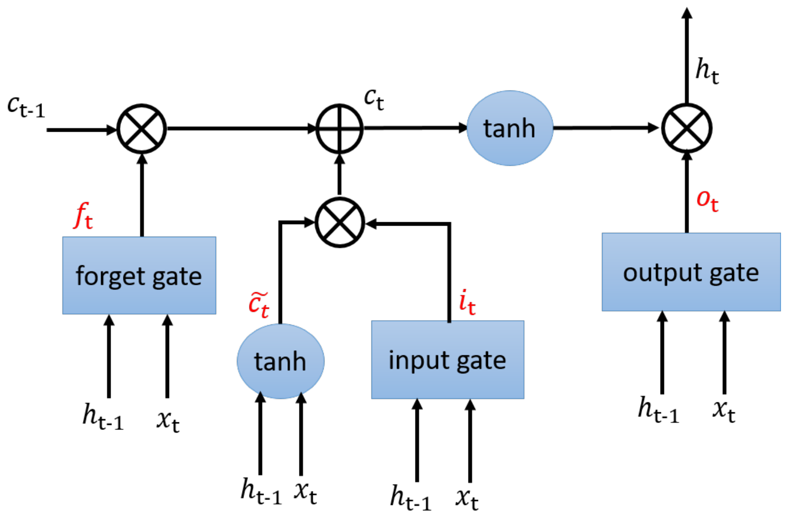


Fig. 2.3: The structure of LSTM cell

LSTM has three gates: input gate i_t, forget gate f_t and output gate o_t. All gates are generated by a sigmoid function over the ensemble of input x_t and the preceding hidden state h_{t-1}. In order to generate the hidden state at current step t, it first generates a temporary result $\widetilde{c_t}$ by a tanh nonlinearity over the ensemble of input x_t and the preceding hidden state h_{t-1}, then combines this temporary result $\widetilde{c_t}$ with history c_{t-1} by input gate i_t and forget gate f_t respectively to get an updated history c_t, finally uses output gate o_t over this updated history c_t to get the final hidden state h_t [62]. After we get final layer output h_t, then a sigmoid function is applied to get the final output y_t.

The LSTM transition equations are the following:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.12}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.13}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.14}$$

$$\widetilde{c_t} = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.15}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \tag{2.16}$$

$$h_t = o_t \circ \sigma_h(c_t), \tag{2.17}$$

where the initial values are $c_0 = 0$, $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product); the subscript t indexes the time step; σ_g is sigmoid function, σ_c hyperbolic tangent function, σ_h is hyperbolic tangent function or, as the peephole LSTM paper [63] suggests, $\sigma_h(x) = x$. Intuitively, the forget gate controls the amount of which each unit of the memory cell is erased, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state [64].

- **Gated Recurrent Units (GRU).** GRU is a gating mechanism in recurrent neural networks, which was introduced in 2014 by Kyunghyun Cho et al [65]. GRU is a variant of LSTM, it is easier to train. It has fewer parameters than LSTM, which only includes two gates, the reset, and the update gate. Fig 2.4 shows the structure of the GRU cell.

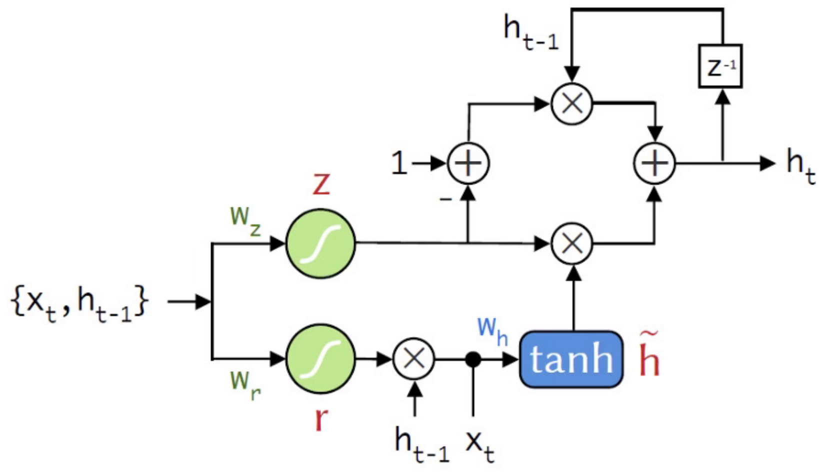


Fig. 2.4: The structure of the GRU cell (Source: from [3])

GRU models text x as follows,

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.18}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2.19}$$

$$\widetilde{h_t} = \phi_h(W_h x_t + U_h(r_t \circ h_{t-1} + b_h)) \tag{2.20}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \widetilde{h_t}, \tag{2.21}$$

where x_t is input vector, h_t is output vector, $\widetilde{h_t}$ is candidate activation vector, z_t is update gate vector, r_t is reset gate vector, W, U and b are parameter matrices and vector; σ_g is sigmoid function, ϕ_h is hyperbolic tangent. After we get layer output h_t, then a sigmoid function is applied to get the final output y_t.

- **Bidirectional LSTM.** Bidirectional LSTM is an extension of traditional LSTM that can improve model performance on sequence classification problems. Bidirectional LSTM trains two LSTMs on the input sequence. Firstly, it computes the forward hidden sequence $\overrightarrow{h_t}$; then, it computes the backward hidden sequence $\overleftarrow{h_t}$; finally, it combines $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ to generate the hidden state h_t. With this architecture, both past context and future context of a specific word can be made use of. This can provide additional context to the network and result in faster and even fuller learning on the problem. Fig 2.5 shows the architecture of Bidirectional LSTM with three consecutive steps. A Bidirectional LSTM is implemented by the following functions:

$$\overrightarrow{h_t} = \sigma_h(W_{\overrightarrow{h}} x_t + W_{\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{2.22}$$

$$\overleftarrow{h_t} = \sigma_h(W_{\overleftarrow{h}} x_t + W_{\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \tag{2.23}$$

$$h_t = W_{\overrightarrow{h}} \overrightarrow{h_t} + W_{\overleftarrow{h}} \overleftarrow{h_t} + b_h \tag{2.24}$$

After we get final layer output h_t, then a sigmoid function is applied to get the final output y_t. In the following, we denote Equations 2.22 to 2.24 as $h_t = BiLSTM(x_t)$.

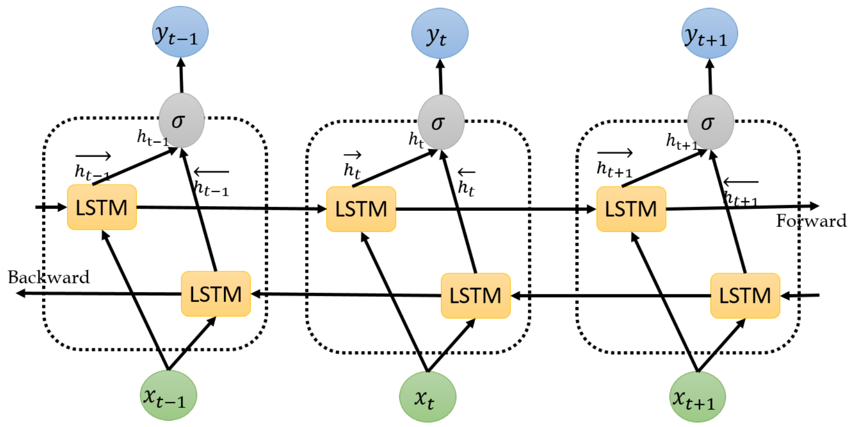


Fig. 2.5: The architecture of Bidirectional LSTM with three consecutive steps

- **Loss Function.** The parameters of the networks we described above are trained to minimize the cross-entropy of the prediction and ground truth.

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)], \tag{2.25}$$

where y_i is the ground truth label, $\hat{y}_i$ is prediction probabilities, and N denotes the number of training samples.

2.3 Experiments

2.3.1 Dataset Description

We evaluate the proposed models on three annotated datasets that are from three social media platforms, Twitter, Wikipedia, and White Supremacy Forum. In this section, we provide a basic introduction for all of them. Table 2.1 shows the detailed statistics of the three datasets.

Table 2.1: Statistics of the three datasets

Dataset	Total	Classes Distribution
Twitter	24783	Offensive 19190 Hate_speech 1430 Neither 4163
White Supremacy Forum	42812	Not hate 38028 Hate 4784
Wikipedia	159571	Toxic 15294 Not toxic 144277

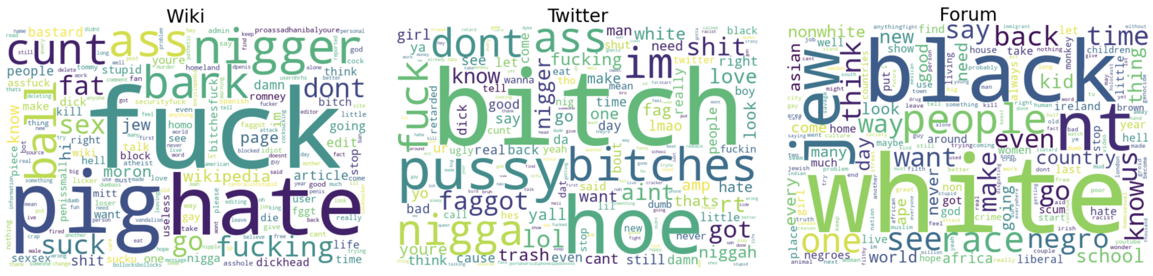
- **Twitter.** The Twitter dataset includes 25K annotated tweets made available by the authors of [24]. In this dataset, 1430 tweets are labeled as hate speech, 19190 tweets are labeled as offensive language, and the remaining (4163) tweets are marked as harmless (neither hate speech nor offensive language). In our experiments, we merge the offensive language labeled tweets and hate speech labeled tweets into one positive class for binary classification.

- **Wikipedia.** Wikipedia dataset includes 160k Wikipedia comments which have been labeled by human raters for toxic behavior. In this dataset, 15294 comments are labeled as toxic, and the remaining comments are marked as not toxic, the dataset is provided by [66].

- **White Supremacy Forum.** Forum dataset includes 42k annotated forums made available by [67]. In this dataset, 4784 forums are labeled as hate speech, 38028 forums are labeled as not hate speech.

2.3.2 Word Cloud

A word cloud is a powerful visual representation object for text processing, in which the sizes of words shows the frequency and importance of specific words. The bigger size of the word means it's more important. Figure 2.6 shows the word clouds of the positive classes of the three datasets. From the three word clouds, we can notice that the hate speech type of the Wikipedia dataset and Twitter dataset are general and similar to each other, while the Forum dataset is focusing on race orientated hate speech.



(a) Wikipedia (b) Twitter (c) Forum

Fig. 2.6: Word cloud of three datasets

2.3.3 Pre-Processing

Since the data we got is from social media platforms, the raw texts usually include a lot of noise, like punctuation, URLs that will affect our experiment result, we pre-process our text data at the very first step.

- **Remove Noise.** Remove the URLs, punctuation marks.
- **Lowercasing.** Lowercasing all the texts, which avoids capitalized versions of words being treated as separate features to lowercase versions of the same word.
- **Stop Words Removal.** A stop word is a commonly used word (such as "the", "a", "an", "in") that a search engine has been programmed to ignore. We would not want these commonly used words taking up space in our database, or taking up the valuable processing time. Here, we remove the stop words, then we can focus on the important words instead.
- **Tokenization.** Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each

of these smaller units is called tokens. In this thesis, we split all the sentences into words during the tokenization process.

- **Token normalization.** Two commonly used techniques for text normalization are *Stemming* and *Lemmatization.* Stemming is a process of removing and replacing suffixes to get to the root form of the word, which is called the *stem*. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis. Lemmatization technique returns the base or dictionary form of a word, which is known as the lemma. Table 2.2a shows how stemming works. Table 2.2b shows how lemmatization works. In our experiments, we use lemmatization as the token normalization method.

Table 2.2: Stemming and lemmatization

(a) Stemming

Original_word	Stemmed_word
feet	feet
cats	cat
wolves	wolv
talked	talk

(b) Lemmatization

Original_word	Lemmatized_word
feet	foot
cats	cat
wolves	wolf
talked	talk

Text Feature Extraction

While applying machine learning algorithms, the raw text data cannot be fed directly to machine learning algorithms. Most algorithms accept numerical feature vectors with a fixed size but can not deal with the raw text data with variable length. To address this problem, we need to extract numerical features from text content. In traditional machine learning, we usually convert a collection of text documents to a matrix of token counts by using Bag of Words or Term Frequency-Inverse Document Frequency.

- **Bag of Words (BOW).** BOW is a method that is to extract numerical features from text content. Firstly, **tokenizing** all sentences and store all unique tokens that generated from all sentences into a dictionary (a big bag); secondly, **counting** the occurrences of tokens in each sentence. In this case, we convert our raw sentences into feature vectors. Thus we can represent a corpus of sentences as a matrix with one row per sentence and one column per token occurring in the corpus. For each token, we will have a feature column, this is called **text vectorization**. Table 2.3 show a simple example of convert three sentences into three feature vectors. Assuming we have three sentences show in Table 2.3a, firstly we can get the first row of Table 2.3b, which contains all the unique tokens that occurs in the three sentences, then we count the occurrences of tokens in each sentence to get the feature vector of each sentence. The general process of converting a collection of text sentences into numerical feature vectors is **vectorization**. By using BOW, sentences are described by word occurrences.

Table 2.3: Bag of words

(a) Sentences

good movie
not a good movie
did not like

(b) Feature vectors

good	movie	not	a	did	like
1	1	0	0	0	0
1	1	1	1	0	0
0	0	1	0	1	1

- **Term Frequency-Inverse Document Frequency (TF-IDF).** In a large text corpus, some words occur very frequently (e.g. “the”, “a” in English) while carrying very little meaningful information about the actual contents. In order to focus on more important information, it is common to use the TF-IDF transform. By using TF-IDF transform, we can can re-weight the count features into floating-point values suitable for usage by classifier. The TF-IDF score is calculated by the following formula:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D), \tag{2.26}$$

where $tf(t,d)$ represents the frequency for term t in document d (Table 2.4 shows the variants of $tf(t,d)$), and idf(t, D) represents the inverse document frequency

$$idf(t,D) = log\frac{N}{|\{d \in D : t \in d\}|}, \tag{2.27}$$

where $N = |D|$ represents the total number of documents in corpus, and $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears.

Table 2.4: Term frequency ($tf(t,d)$)

Weighting scheme	TF weight
Binary	0,1
Raw count	$f_{t,d}$
Term frequency	$f_{t,d}/\sum_{t' \in d} f_{t',d}$
Log normalization	$1 + log(f_{t,d})$

A high weight in TF-IDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents.

2.3.4 Evaluation Metrics

In general, classifiers that show higher accuracy scores represent better performance in classification tasks. However, the class distribution of the three datasets that we used in our experiments are very imbalanced, using accuracy score is misleading. Here, we use macro averaged precision, recall, and F1 score to summarize models' performance, which may provide a more informative evaluation strategy for imbalanced classification models [68].

Accuracy, which is the proportion of correct predictions, is suitable for evaluating models based on balanced datasets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.28}$$

where TP (True Positive) is the number of texts that are positive and predicted correctly as positive; TN (True Negative) is the number of texts that are negative and predicted correctly as nagetive; FP (False Positive) is the number of texts that are negative but predicted incorrectly as positive; FN (False Negative) is the number of texts that are positive but predicted incorrectly as negative.

Precision, which is also referred to as positive predictive value (PPV), is the proportion of positive results that are truly positive:

$$Precision = \frac{TP}{TP + FP}. \tag{2.29}$$

Recall, which is also referred to as the true positive rate or sensitivity, is the ability of a test to correctly identify positive results to get the true positive rate:

$$Recall = \frac{TP}{TP + FN}. \tag{2.30}$$

F1 score, which is also referred to as the F score or F measure, is defined as the weighted harmonic mean of the precision and recall such that the best score is 1.0 and the worst is 0.0. F1 score is calculated as follow:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{2.31}$$

2.3.5 Experimental Results

We conduct hate speech detection on Twitter, Wikipedia, and White Supremacy Forum datasets that we have described in Table 2.1. For each dataset, we assign 75 percent of the whole dataset for training, and the test set contains the remaining 25 percent. Table 2.5 shows the size of the training and test sets for three datasets. In addition, for deep learning methods, we assign 10 percent of the training set to the validation set. Each model is trained using the training set, and then the test set is used to measure the performance of the model.

Table 2.5: The size of training and test datasets

Dataset	Training size(75%)	Test size(25%)
Twitter	18587	6196
Forum	32109	10703
Wikipedia	119678	39893

Tables 2.6, 2.7, and 2.8 show the macro averaged precision, recall and F1 score for hate speech detection results on three datasets respectively. The macro average precision, recall, and F1 score indicate arithmetic averages of the per-class precision, recall, and F1-score to show the overall performance of all classes. The best precision, recall, and F1 score results of different models for each dataset are identified by underlining and bolding.

Table 2.6: Macro averaged precision, recall and F1 score for Twitter Dataset

Method	Precision	Recall	F1 score
Traditional Machine Learning Methods			
Naïve Bayes & TFIDF	0.9033	0.5550	0.5581
Naïve Bayes & Count	0.9053	0.7757	0.8215
SVM & TFIDF	0.9027	0.9058	0.9042
SVM & Count	0.9065	0.9102	**0.9083**
Logistic Regression & TFIDF	**0.9113**	0.8532	0.8787
Logistic Regression & Count	0.9054	**0.9113**	**0.9083**
Deep Learning Methods			
SimpleRNN	0.8889	0.8637	0.8756
LSTM	0.8937	0.8656	0.8788
GRU	0.8947	0.8738	0.8838
Bidirectional LSTM	0.8972	**0.8753**	0.8858
CNN	**0.9105**	0.8745	**0.8912**

It is clear from Table 2.6 that for Twitter dataset, traditional machine learning methods perform better than deep learning methods. Among all traditional machine learning methods, Logistic Regression with the TFIDF feature shows the best precision while Logistic Regression with Bag of Words feature shows the best recall and F1 score. Among all deep learning methods, CNN shows the best precision and F1 score while Bidirectional RNN shows the best recall.

Table 2.7: Macro averaged precision, recall and F1 score for Forum Dataset

Method	Precision	Recall	F1 score
Traditional Machine Learning Methods			
Naïve Bayes & TFIDF	0.9319	0.5986	0.6399
Naïve Bayes & Count	0.8767	0.8750	0.8758
SVM & TFIDF	0.9238	0.6641	0.7231
SVM & Count	**0.9712**	**0.9081**	**0.9367**
Logistic Regression & TFIDF	0.8977	0.6486	0.7025
Logistic Regression & Count	0.9672	0.8868	0.9221
Deep Learning Methods			
SimpleRNN	0.9924	0.9853	0.9888
LSTM	0.9858	0.9894	0.9876
GRU	0.9891	0.9891	0.9891
Bidirectional LSTM	0.9824	0.9882	0.9853
CNN	**0.9944**	**0.9933**	**0.9939**

Table 2.7 shows that for the Forum dataset, deep learning methods perform better than traditional machine learning methods. Among all traditional machine learning methods, SVM with TFIDF feature achieves the best precision, recall, and F1 score. Among all deep learning methods, CNN achieves the best precision, recall, and F1 score.

Table 2.8: Macro averaged precision, recall and F1 score for Wikipedia Dataset

Method	Precision	Recall	F1 score
Traditional Machine Learning Methods			
Naïve Bayes & TFIDF	**0.9554**	0.5714	0.6031
Naïve Bayes & Count	0.9117	0.7617	0.8159
SVM & TFIDF	0.9546	0.7239	0.7920
SVM & Count	0.9165	0.8089	0.8526
Logistic Regression & TFIDF	0.9362	0.7902	0.8450
Logistic Regression & Count	0.9111	**0.8242**	**0.8611**
Deep Learning Methods			
SimpleRNN	0.5421	0.5372	0.5393
LSTM	0.8600	0.8357	0.8473
GRU	0.8683	0.8316	0.8487
Bidirectional LSTM	**0.8790**	0.8281	0.8512
CNN	0.8785	**0.8403**	**0.8581**

Table 2.8 shows that for the Wikipedia dataset, deep learning methods perform better in terms of precision and F1 score than traditional machine learning methods. Among all traditional machine learning methods, Logistic Regression with Bag of Words feature achieves the best recall and F1 score while Naïve Bayes with TFIDF feature achieves the best precision. Among all deep learning methods, CNN achieves the best recall and F1 score while Bidirectional RNN performs the best precision.

2.4 Summary

In this chapter, we conduct hate speech detection on three different social media platforms by using widely-used traditional machine learning models and deep learning models based on rich amount of annotated data. The experimental results show that these models can achieve decent performance.

CHAPTER 3
Hate Speech Detection via a Pre-trained Language Model

In previous work, we conduct hate speech detection based on rich annotated data and get decent performance. However, in real life, data labeling is very time and effort-consuming. If we want to conduct hate speech detection on a social media platform that only limited labeled data is available for us, then how can we tackle this challenge?

3.1 Introduction

In Chapter 2, we have shown the good performance of applying deep learning models for hate speech detection in rich resource datasets. However, modern deep learning models suffer data-hungry and computationally expensive problems, good performance usually depends on sufficient amounts of labeled data. Recently, pre-trained language models have attracted massive interest in the NLP field, which aims at finding the best methods for word or text representations. The key idea behind pre-trained language models is to train a large generative model on a huge size corpus, then use the resulting representations on tasks for which only limited amounts of labeled data are available. Pre-trained language models can be regarded as a black box that has a good understanding of natural language, which can be easily applied and fine-tuned to deal with many NLP tasks. In this case, we don't have to train a new model from scratch. Unsupervised pretraining of language models on large corpora such as ELMo [16], GPT [17] and BERT [6] significantly improve performance on many NLP tasks. In this chapter, considering the low-resource setting scenario, we propose to use a pre-trained Bidirectional Encoder Representations from Transformers (BERT) model to address the low resource issue for hate speech detection.

3.2 Methodology

3.2.1 BERT

BERT, as a powerful pre-trained language model, is built based on Transformer encoder [69]. BERT is designed to train deep bidirectional representations from unlabeled texts by considering both left and right contexts [6].

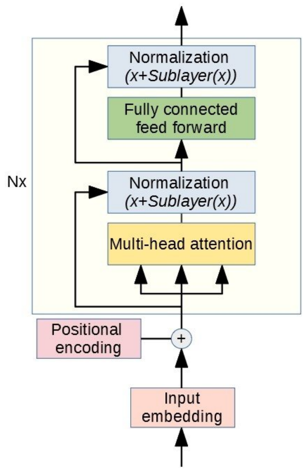


Fig. 3.1: Transformer encoder architecture (Source: from [4])

The multi-layer bidirectional transformer encoder (see Fig 3.1) contains a stack of identical layers. In each layer, there are two sub-layers, a multi-head self-attention mechanism and a simple position-wise fully connected feed-forward network. It employs a residual connection [70] around both sub-layers, followed by a layer normalization [4]. In this case, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer [69].

For the multi-head self-attention mechanism (see Fig 3.2), firstly, a scaled dot-product attention is defined as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{3.1}$$

where Q is the matrix of queries, K is the matrix of keys, V is the matrix of values and d_k is the dimension of the Q and K matrices. Then, a multi-head attention is defined as:

$$MuitiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O, \tag{3.2}$$

where $head_i = Attention(QW_i{}^Q, KW_i{}^K, VW_i{}^V)$. Multi-head attention works on projecting the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v(dimension of the values matrix), respectively. Then, for each projected version of the queries, keys and values, attention function is performed in parallel, yielding in $d_v - dimentional$ output values. Finally, these are concatenated and projected, resulting in the final values [69]. The key point behind self-attention is that all of the keys, values and queries come from the same place.

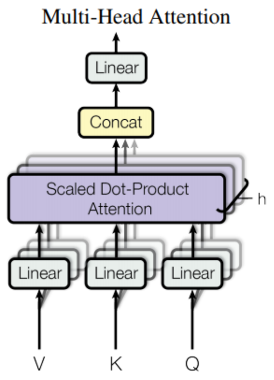


Fig. 3.2: Multi-head attention (Source: from [5])

Pre-training BERT

For the BERT model framework, there are two important steps: pre-training and fine-tuning [6]. During the pre-training process, the model is trained on the English Wikipedia and the Book Corpus using the following tasks:

Task #1: Masked LM. Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-right model [6]. However, the previous unidirectional language models are only considering context from left-to-right or right-to-left while learning general language representations. Since bidirectional conditioning enables each word to indirectly "see itself", *Masked LM* is proposed to train a deep bidirectional representation.

At first, a ratio of tokens in the input sequence is masked out at random. Then, the entire sequence is fed to a deep bidirectional Transformer [69] encoder. The objective of the model is to predict the original vocabulary id of the masked words only based on their context. In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM [6]. By using this "masked language model" pre-training process, BERT alleviates the unidirectional constraint, which enables the representation to fuse the left and right context. The following example illustrates how masked LM works.

- Input = That's [MASK] she [MASK]
- Output = That's what she said

Task #2: Next Sentence Prediction (NSP). A binarized *NSP* task is pre-trained to understand the relationship between sentences, which can be easily generated from any monolingual corpus. Given a pair of two input sentences A and B, and then the model learns to classify whether or not sentence B follows sentence A. Specifically, while choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time B it is a random sentence from the corpus (labeled as NotNext) [6]. The following examples illustrate how NSP works.

- Input = [CLS] the man went to [MASK] store [SEP]
  he bought a gallon [MASK] milk [SEP]
  Label = IsNext

- Input = [CLS] the man [MASK] to the store [SEP]
  penguin [MASK] are flight ##less birds [SEP]
  Label = NotNext

After pre-trained based on the above two self-supervised tasks, the model is initialized with the pre-trained parameters. Since the BERT model is pre-trained on general corpora, and the hate speech detection task that we are dealing with is related to social media content. Then we have to fine-tune it using annotated hate speech datasets after analyzing the contextual information extracted from BERT pre-trained layers. For fine-tuning, all the parameters are fine-tuned using labeled data for specific tasks. In our experiments, all the parameters are fine-tuned for hate speech detection.

The layers of BERT architecture are showed in Fig 3.3, where E_n defined the n-th token in the input sequence, Trm defines the Transformer block, and T_n defines the corresponding output embedding. We can see that it is not the same as traditional sequential or recurrent models, the attention architecture processes all the input tokens in parallel. Pre-trained BERT model can be fine-tuned with just one additional layer to obtain state-of-art results in a wide range of NLP tasks [6].

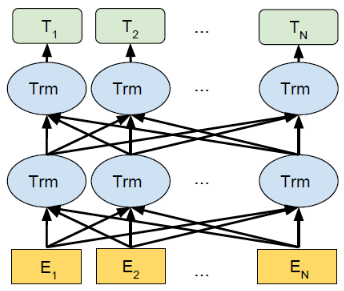


Fig. 3.3: BERT architecture (Source: from [6])

3.2.2 BERT Based Hate Speech Detection Model

Fig 3.4 shows the overall architecture of our BERT based model. Firstly, we remove punctuations, URLs, stopwords, and lowercase all the texts. Secondly, we tokenize texts by padding them into a max length and add the [CLS] and [SEP] tokens at the appropriate positions. Thirdly, we compute the sequence embedding from BERT. Then we apply dropout with a probability factor of 0.3 to regularize and prevent overfitting. Finally, we apply a softmax classification layer for prediction. The softmax layer is a fully connected neural network layer with the softmax activation function $\sigma : \mathcal{R}^K \rightarrow \mathcal{R}^K$:

$$\sigma(Z)_i = \frac{e^{z_i}}{\Sigma_{j=1}^{K} e^{z_j}}, \tag{3.3}$$

where $i = 1, ..., K$, and $z = (z_1, ..., z_K) \in \mathcal{R}^K$ is the intermediate output of the softmax layer (also called logits). The output node with the highest probability is then chosen as the predicted class label for the input.

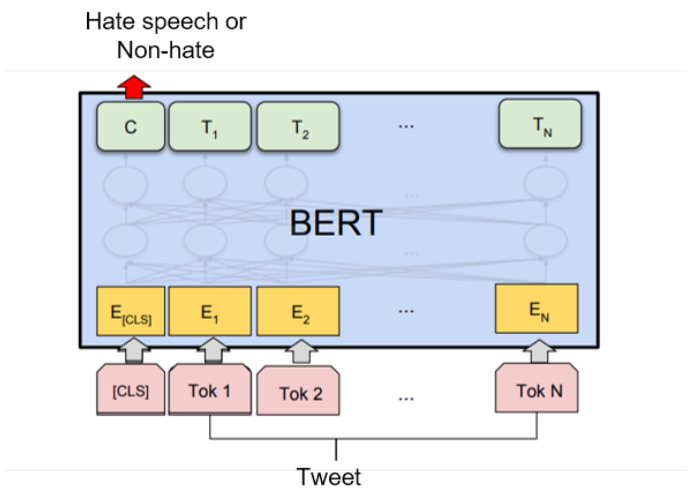


Fig. 3.4: Bert based hate speech detection model (Source: from [6])

3.3 Experiments

3.3.1 Dataset Description

The datasets we use in our experiments are still from Twitter, Wikipedia, and White Supremacy Forum datasets that we have introduced in Chapter 2. For each dataset, we use 600 of the whole dataset for training, and the test set contains the remaining of each dataset. Table 3.1 shows the size of the training and test sets for our experiments in this chapter.

Table 3.1: The size of training and test datasets

Dataset	Training size	Test size
Twitter	600	24183
Forum	600	42212
Wikipedia	600	158971

3.3.2 Pre-Processing

We remove stop words, remove punctuation marks, URLs, and convert all texts to lower case. We also use the lemmatization technique as the token normalization method. The details of these pre-processing steps have been introduced in Chapter 2.

3.3.3 Experimental Results

Every model is trained using the training set and then the test set is used to measure the performance of the model. For deep learning methods, we assign 100 data points of the training set to the validation set. In this chapter, the experiments using different datasets with the same size of the training datasets, the evaluation metrics that we use here are the macro averaged precision, recall, and F1 score. Table 3.2 shows the macro averaged precision, recall, and F1 score for three datasets respectively. Here, we compare the performances of BERT based model with traditional machine learning models and deep learning models based on the small-size training datasets. The best precision, recall, and F1 score results of different models for each dataset are identified by underlining and bolding.

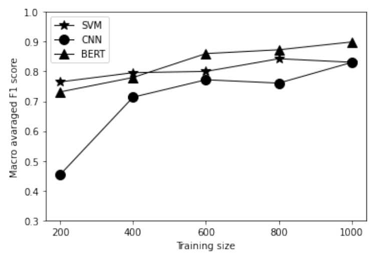
Table 3.2: Macro average of precision, recall and F1-score for three datasets

Datasets	Models	Precision	Recall	F1-score
Twitter Dataset	Naïve Bayes	0.8840	0.6627	0.7091
	SVM	0.8484	0.7704	0.8014
	Logistic Regression	0.8532	0.7128	0.7561
	Simple RNN	0.5488	0.5358	0.5382
	LSTM	0.4156	0.5000	0.4539
	Bidirectional LSTM	0.4156	0.5000	0.4539
	GRU	0.4156	0.5000	0.4539
	CNN	0.8000	0.7580	0.7762
	BERT	**0.8865**	**0.8389**	**0.8602**
Forum Dataset	Naïve Bayes	0.6753	0.5504	0.5634
	SVM	0.6551	0.5816	0.6006
	Logistic Regression	0.8175	0.5357	0.5395
	Simple RNN	0.5805	0.5002	0.4710
	LSTM	0.5813	0.5820	0.5816
	Bidirectional LSTM	0.8401	0.5123	0.4956
	GRU	0.7444	0.5006	0.4718
	CNN	**0.8879**	0.5159	0.5024
	BERT	0.7385	**0.6338**	**0.6656**
Wikipedia Dataset	Naïve Bayes	0.8998	0.6176	0.6673
	SVM	0.7236	0.6541	0.6801
	Logistic Regression	0.8935	0.5648	0.5919
	Simple RNN	0.7646	0.5002	0.4752
	LSTM	0.9102	0.5028	0.4806
	Bidirectional LSTM	**0.9198**	0.5042	0.4834
	GRU	0.6437	0.5018	0.4791
	CNN	0.6724	0.6273	0.6449
	BERT	0.8912	**0.7685**	**0.8155**

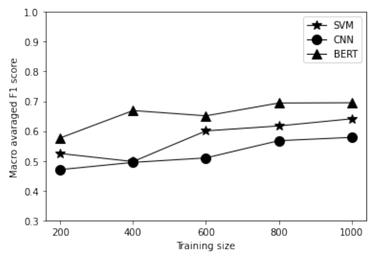
It is clear from Table 3.2 that, for Twitter dataset, BERT shows the best precision, recall and F1 score; for Forum dataset, CNN shows the best precision, BERT shows the best recall, and F1 score; for Wikipedia dataset, Bidirectional LSTM shows the best Precision, BERT shows the best recall, and F1 score. Overall, for three datasets, BERT always shows the best recall and F1 score among the deep learning models and traditional machine learning models used in our experiments.

Besides comparing the BERT with the deep learning models and traditional machine learning models by using a fixed size of the training dataset, we also compare the different performances of SVM, CNN, and BERT by using different sizes (200, 400, 600, 800, 1000) of the training dataset. The reason why we choose CNN and SVM to compare with BERT is that, according to Table 3.2, SVM shows the highest F1 score among traditional machine learning models, CNN shows the highest F1 score among deep learning models.

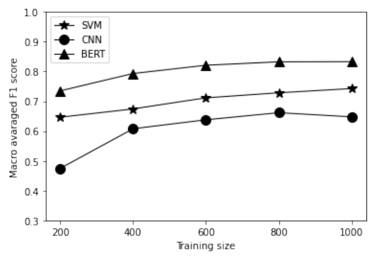
We simply choose 200, 400, 600, 800, 1000 from the original Twitter dataset, Forum dataset, and Wikipedia dataset for training, and the remaining for testing. The evaluation metric we use here is the Macro average of the F1-score. Fig 3.5 shows that, for Forum and Wikipedia datasets, BERT always shows the best performance in F1 score. For Twitter dataset, by using a small size (200, 400) training dataset, SVM can get a decent performance, even better than using BERT. However, when the training size is a little bit larger, BERT shows the best performance.



(a) Twitter Dataset



(b) Forum Dataset



(c) Wikipedia Dataset

Fig. 3.5: Macro averaged F1 score of three datasets with different training size

3.4 Summary

In this chapter, we conduct hate speech detection on three different social media platforms based on limited labeled data. We fine-tune the BERT model on each social media platform to detect hate speech. The experimental results show that by using BERT, we can achieve better performance than widely-used machine learning models while only limited labeled data is available.

CHAPTER 4
Multitask Learning for Hate Speech Detection

In previous work, we fine-tune a language model to conduct hate speech detection on single task learning based on limited labeled data for each social media platform. If we have limited labeled data from several related social media platforms, can we utilize the correlations among platforms to improve hate speech detection on all the platforms?

4.1 Introduction

In Chapter 2, we apply the traditional machine learning and deep learning models to detect hate speech based on single-task supervised objective functions. By using large sizes of annotated datasets to train our models, we do get good performances. However, in reality, the data that can be used for hate speech detection is limited, it is not easy to collect enough annotated data to analyze the models. Then in Chapter 3, we propose to fine-tune the pre-trained BERT model which employs a pre-trained embedding to map words into vectors with semantic implications. By using the pre-trained BERT model, we can get decent performance by just using a relatively small size of training data. However, BERT is trained on a large number of words and articles from Wikipedia and the Book Corpus. Therefore, this method introduces extra knowledge from the large-size corpus. In this chapter, we propose to use the multitask learning technique to leverage potential correlations among related tasks to extract common features. By using multitask learning, we can not only increase corpus size implicitly but also absorb knowledge from related tasks. Multitask learning is defined as "an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias" by Caruana [18]. Multitask learning has been applied to many different domains. The substantial benefit of using extra tasks has been validated [18]. One of the most important benefits of multitask learning is that it provides an approach to access resources developed for similar tasks. By learning related tasks jointly while using a shared representation, we can get access to larger amounts of usable data. And with the help of other tasks, the performance of a single task can be improved. In this chapter, considering the low-resource setting scenario, we propose multitask learning to detect hate speech in different online platforms.

4.2 Methodology

4.2.1 Multitask Learning

The key idea of multitask learning is to solve multiple related tasks simultaneously. Recently, the neural network-based models for multitask learning have been employed in computer vision and natural language processing, and the benefits have been empirically validated [71–74]. Especially, given K learning tasks $\{T^k\}_{k=1}^{K}$, where all the tasks or subset of them are related, multitask learning aims to improve the performance of a model for T^k by using the knowledge contained in all or some of the K tasks [75].

In our scenario, D^k is defined as an online platform with N_k texts for task k, i.e.,

$$D^k = \{(x_i^k, y_i^k)\}_{i=1}^{N_k}, \tag{4.1}$$

where x_i^k and y_i^k denote a sentence and corresponding label for task k.

4.2.2 Multitask Learning for Hate Speech Detection

We adopt bidirectional LSTM (BiLSTM) to model the text data, which consists of two LSTM networks to propagate text forward and backward to capture the dependencies among words. Previous work in Chapter 3 also shows relatively good performance by using BiLSTM while we conduct hate speech detection based on limited labeled data. In this chapter, we still consider using multitask learning to detect hate speech based on several limited labeled datasets from different social media platforms.

Architecture of Multitask Learning

Fig 4.1 shows the multitask learning architecture. It shares the hidden layers between all tasks and keeps several task-specific output layers. In our case, the embedding layer and Bidirectional LSTM layer are shared to extract the common features for all tasks, then fed into the corresponding task-specific output layer for classification. For example, given three hate speech detection tasks k_1, k_2 and k_3 that from different social media platforms (Twitter, Wikipedia, Forum), it takes the view that the features of each hate speech detection task can be shared by the other two hate speech detection tasks. Given K supervised hate speech detection tasks, $T^1, T^2, ..., T^K$, a jointly learning model is trained to transform multiple inputs into a combination of predicted distributions in parallel.

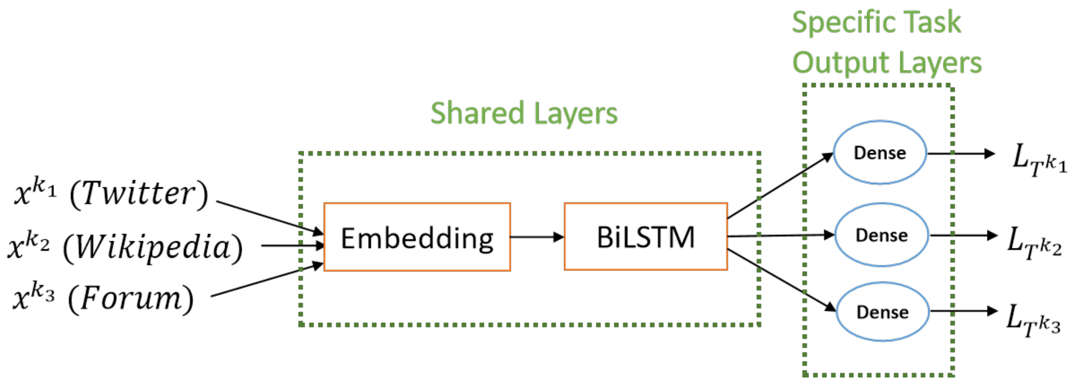


Fig. 4.1: Architecture of multitask learning

For a sentence x_i^k in task T^k, the feature h_i^k will be extracted by the shared layers, in this case, we can firstly get its shared representation h_i^k.

$$h_i^k = BiLSTM(x_i^k), \tag{4.2}$$

After the feature h_i^k is derived by the multitask architecture, it will be fed into the corresponding task-specific dense layer for classification. The overall training objective is to minimize the cross-entropy of predicted and true distributions on all the tasks.

$$\hat{y}_i^k = \sigma(Wh_i^k + b), \tag{4.3}$$

$$L_{T^k} = -\frac{1}{N}\sum_{i=1}^{N}[y_i^k log(\hat{y}_i^k) + (1 - y_i^k)log(1 - \hat{y}_i^k)], \tag{4.4}$$

$$L = \sum_{k=1}^{K} L_{T^k}, \tag{4.5}$$

where $\hat{y}_i^k$ is the prediction of a input sentence in task k, y_i^k is the ground truth of a input sentence in task k, N is the number of training samples in specific task, L_{T^k} is the loss of task k, L is the total loss of all tasks.

4.3 Experiments

4.3.1 Dataset Description

In this chapter, we compare the performances of single-task (BiLSTM model) with multitask learning for each dataset. We use balanced datasets to conduct our experiments. To generate balanced datasets for our experiments, we under-sample the majority class (negative class). Table 4.1 shows the class distributions for the three datasets we use in this chapter. In our experiments, we choose 100, 200, 300, 400, and 500 data points from the three balanced datasets for training. We also assign another 100 data points for validation, and the remaining for testing.

Table 4.1: The classes distribution for three datasets

Dataset	Positive	Negative
Twitter	4000	4000
Forum	4700	4700
Wikipedia	15000	15000

4.3.2 Pre-Processing

We remove stop words, punctuation marks, URLs, and convert all texts to lower case. We also use the lemmatization technique as the token normalization method. The details of these pre-processing steps have been introduced in Chapter 2.

4.3.3 Training Details

The training steps of our multitask learning are different from single-task learning. In multitask learning, the labeled data for training each task should come from completely different datasets. According to [71], the training is achieved in a stochastic manner by looping over the tasks:

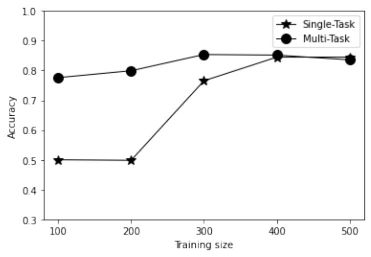
1. Select a random task.

2. Select random training examples from this task. In our experiments, we set a batch size 32, and then randomly select a batch from the specific task as training samples.

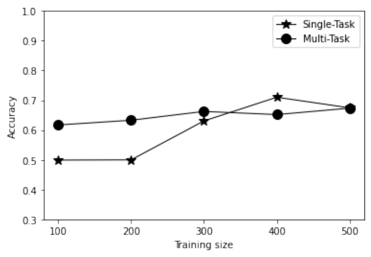
3. Update the parameters for this task by taking a gradient step with respect to these examples.

4. Go to 1.

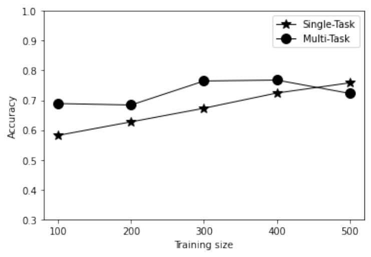
4.3.4 Experimental Results



(a) Twitter Dataset



(b) Forum Dataset



(c) Wikipedia Dataset

Fig. 4.2: Accuracy of three datasets with different training size

Since we use balanced datasets to conduct experiments, the evaluation metric here we use is Accuracy. Figure 4.2 shows the result of our experiments. It is clear from Fig 4.2

that, for three datasets, the performances of using multitask learning are much better than just training our model on a single dataset when we only use 100, 200, 300 samples for training. However, while the size of our training set is larger, the performances of using multitask learning are not better than just training our model on a single dataset. This means that in multi task learning, the quality of data (how close to the target data) is much more important than the quantity (how many related tasks data).

4.4 Summary

In this chapter, we develop a multitask learning model to conduct hate speech detection on three social media platforms in parallel. The experimental results show that, by leveraging the knowledge from the three related hate speech detection tasks, the performances of hate speech detection on each single social media platform can be improved when only limited labeled data are available.

CHAPTER 5

Domain Adaptation for Hate Speech Detection

In previous work, we implement hate speech detection based on labeled data with large size of training datasets and small size of training datasets, while generating labeled data is always an obstacle, can we achieve hate speech detection when the labeled data of our target social media platform is not available?

5.1 Introduction

Over the past few years, significant success has been validated in classification areas by using traditional machine learning technologies. However, while applying machine learning methods for a new machine learning task, the cost of generating labeled data is often challenging. Costly annotation limits the further development of machine learning approaches. In this case, domain adaptation approaches have been introduced to address the lack of labeled data problems. The task of domain adaptation is to leverage sufficient annotated data from the related source domain to the unlabeled target domain. In many cases, we have enough labeled training data for the source domain, we wish to learn a classifier from the source domain and then apply it into our unlabeled target domain with different distributions. By transferring invariant structures or features from the source domain to the target domain, we can alleviate the distribution discrepancy of different domains. According to [19], a good representation transfer is one for which an algorithm cannot learn to identify the domain of origin of the input observation. To achieve domain adaptation, then in [76], the authors proposed a neural network algorithm, named Domain Adversarial Neural Network. Domain Adversarial Neural Network architecture aims at encouraging the network's hidden layer to learn a representation that is predictive of the source example labels, but uninformative about the domain of the input. The effectiveness of applying Domain Adversarial Neural Network has been validated from extensive experiments on different applications. Domain Adversarial Neural Network architecture develops an explicit pair of the source and target domains that the knowledge can be transferred from one source domain to the target domain. Recently, research in [77] propose that domain adaptation can be applied on multiple source domains. In this case, more knowledge might be absorbed from multiple source domains compared with learning from a single-source domain. Considering the zero-resource settings of the target domain, we propose to adopt Domain Adversarial Neural Network architecture to detect hate speech on newly developed social media platforms.

5.2 Methodology

5.2.1 Domain Adaptation

Domain adaptation is described by [76] as follows: $\mathcal{X}$ is defined as the input space and $\mathcal{Y} = \{0, 1\}$ is the set of labels in classification tasks. Then we have source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$, which are two different distributions over $\mathcal{X} \times \mathcal{Y}$. An *domain adaptation* learning algorithm is then provided with a labeled source sample S drawn *i.i.d* from $\mathcal{D}_S$, and an unlabeled target sample T drawn *i.i.d* from $\mathcal{D}_T$,

$$S = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{m} \sim (\mathcal{D}_S)^m \quad T = \{\mathbf{x}_i^t\}_{i=1}^{m'} \sim (\mathcal{D}_T)^{m'}, \tag{5.1}$$

where m is the number of samples from source domain, m' is the number of samples from target domain. The learning algorithm aims at building a classifier $\eta : \mathcal{X} \to \mathcal{Y}$ with a low target risk

$$R_{\mathcal{D}_T(\eta)} \overset{\text{def}}{=} \underset{(\mathbf{x}^t, y^t) \sim \mathcal{D}_T}{Pr} \Big(\eta(\mathbf{x}^t) \neq y^t\Big), \tag{5.2}$$

while having no information about the labels of $\mathcal{D}_T$. Here the low target risk $R_{\mathcal{D}_T(\eta)}$ that respect to classifier η is defined by the probability that the prediction results $\eta(\mathbf{x}^t)$ do not match the the ground truth y^t.

In our cases, we assume this scenario: for a new social media platform, we want to conduct hate speech detection on this platform. However, we do not have any labeled data for this platform. Then we try to leverage the labeled data from other related social media platforms and apply the domain adaptation technique to address the hate speech detection task on our target social media platform. Here, the hate speech detection on our target social media platform is regarded as a target domain task, and the other related social media platforms with labeled data are regarded as source domains. Our goal is to transfer the knowledge from source domains to our target domain, then to achieve high accuracy on the target hate speech classification task without labeled data of our target domain.

5.2.2 Domain Adversarial Neural Network for Hate Speech Detection

To achieve domain adaptation for hate speech detection, we propose a domain adversarial neural network to learn a model that can generalize well from the source domain to the target domain. The key point behind of domain adversarial neural network is that the source risk is expected to be a good indicator of the target risk when the distributions of the source and the target are similar. Therefore, the feature representation derived from the neural network should not contain any discriminative domain information about the input (source or target). The domain adversarial neural network includes two main key parts: *Source Risk Minimization* and *Domain Adaptation Regularizer*. The source risk minimization is to achieve high accuracy on the hate speech detection of the source domain. The domain adaptation regularizer is to force representations of examples where both the source and the target domain are as indistinguishable as possible.

Source Risk Minimization

For source domain classification, given an input text $\mathbf{x}$, we denote the feature representation of this input as $h(\mathbf{x})$, where $h(\cdot)$ is an CNN model used as the feature extractor.

$$h(\mathbf{x}) = CNN(\mathbf{x}). \tag{5.3}$$

We denote $f(\mathbf{x})$ as conditional probability that the neural network assigns $\mathbf{x}$ to class y, where y is the ground truth label.

$$f(\mathbf{x}) = \sigma(Wh(\mathbf{x}) + b). \tag{5.4}$$

To train the feature extractor $h(\cdot)$ and classifier $f(\cdot)$, we use cross entropy loss as the objective function.

$$\mathcal{L}^s(f(\mathbf{x}), y) = -ylog(f(\mathbf{x})) - (1-y)log(1-f(\mathbf{x})). \tag{5.5}$$

Given a training source sample $S = \{(\mathbf{x}_i^S, y_i^S)\}_{i=1}^m$, to get a low risk on source domain, we need to minimize the objective function:

$$min\left[\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^s(f(\mathbf{x}_i^S), y_i^S)\right]. \tag{5.6}$$

Domain Adaptation Regularizer

To learn representations of examples where both the source and the target domain are as indistinguishable as possible, domain-invariant features need to be learned. We denote the feature representations of unlabeled sample from the target domain $T = \{\mathbf{x}_i^t\}_{i=1}^{m'}$ as $h(T) = \{h(\mathbf{x}_i^t)\}_{i=1}^{m'}$.

We use a domain classifier to classify a given input ($\mathbf{x}^S$ or $\mathbf{x}^t$) is from the source domain $\mathcal{D}_S$ or target domain $\mathcal{D}_T$, we denote the label of target domain as $z = 0$, denote the label of source domain as $z = 1$:

$$o(\phi) = \sigma(d + u^T\phi), \tag{5.7}$$

where $o(\cdot)$ is a domain classifier; ϕ is either $h(\mathbf{x}^S)$ or $h(\mathbf{x}^t)$. Loss function of domain classifier $\mathcal{L}^d$ is defined as

$$\mathcal{L}^d(o(\mathbf{x}), z) = -zlog(o(z)) - (1-z)log(1-o(z)). \tag{5.8}$$

Then the domain classifier is to get low risk on domain classification:

$$min\Big(\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^d(o(\mathbf{x}_i^S),1)+\frac{1}{m'}\sum_{i=1}^{m'}\mathcal{L}^d(o(\mathbf{x}_i^t),0)\Big). \tag{5.9}$$

When the domain classifier is struggling to classify the input is from the source domain or target domain, the feature extractor is to prevent the domain classifier from classifying the input correctly. Finally, the domain classifier will fail, then domain-invariant features can be learned. Here, we use a domain classifier and feature extractor to fight each other to achieve this goal. Then we can get the final objective function as follows.

Final Objective Function

To sum up the *Source Risk Minimization* and *Domain Adaptation Regularizer*, we can get the the final objective function:

$$min\Bigg[\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^s(f(\mathbf{x}_i^S),y_i^S)-\lambda max\Big(\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}^d(o(\mathbf{x}_i^S),1)+\frac{1}{m'}\sum_{i=1}^{m'}\mathcal{L}^d(o(\mathbf{x}_i^t),0)\Big)\Bigg], \tag{5.10}$$

where the first term is to minimize the loss of hate speech detection on the source domain, the second term is to maximize the loss of domain classification task to get domain-invariant features. $\lambda > 0$ weights the domain adaptation regularization term, in our experiments, we set λ as 1.

Domain Adversarial Neural Network

In Equation 5.10, there is a maximization operation, which makes the feature extractor and the domain classifier fight against each other in an adversarial way. Figure 5.1 shows the architecture of our domain adversarial neural network. The domain adversarial neural network includes several major parts: i) feature extractor (the green part), ii) domain classifier (the blue part), iii) label predictor (the yellow part). The feature extractor learns a representation in which the label predictor accurately classifies the hate speech from the source domain, while the domain classifier is unable to distinguish the input example that belongs to the source domain or the target domain. To fool the domain classifier, a special gradient reversal layer (GRL) is inserted between the feature extractor and the domain classifier. During the backpropagation-based training, the gradient reversal layer is used to multiply the gradient by a negative constant. Gradient reversal ensures that the feature distributions over the target domain and the source domain are as indistinguishable as possible for the domain classifier. During the learning process in the domain adversarial neural network, the domain classifier aims to discriminate between the source domain and the target domain, while the feature extractor is adversarially updated to prevent it to succeed. In the end, the domain classifier fails to discriminate the source and target distributions. In this case, the domain-invariant features can be learned.

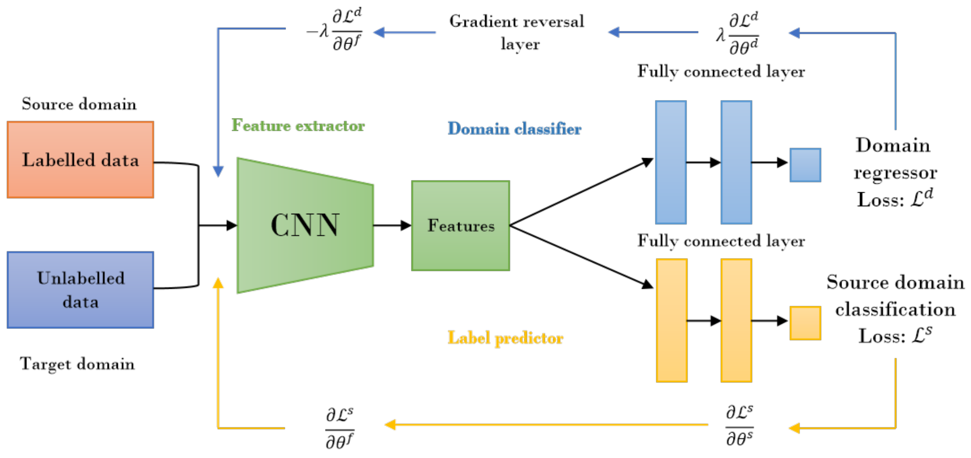

$-\lambda\frac{\partial\mathcal{L}^d}{\partial\theta^f}$
Gradient reversal layer
$\lambda\frac{\partial\mathcal{L}^d}{\partial\theta^d}$
Fully connected layer
Source domain
Labelled data
Feature extractor
Domain classifier
Domain regressor Loss: $\mathcal{L}^d$
CNN
Features
Unlabelled data
Fully connected layer
Source domain classification Loss: $\mathcal{L}^s$
Label predictor
Target domain
$\frac{\partial\mathcal{L}^s}{\partial\theta^f}$
$\frac{\partial\mathcal{L}^s}{\partial\theta^s}$

Fig. 5.1: Architecture of domain adversarial neural network (Source: from [7])

Domain Adversarial Neural Network based on Multiple Source Domains

In the previous description, we are talking about applying a domain adversarial neural network to achieve domain adaptation between one source domain and one target domain. Actually, the domain adversarial neural network can be easily extended to multi-source domain adaptation. We only need to add labeled data from multiple source domains as input, the source risk is the total risk of hate speech detection from multiple source domains. We can achieve domain adaptation by using multiple source domains and transfer the knowledge from multiple source domains to our target domain. For example, in our cases, if we want to conduct hate speech detection on the Twitter platform without label information, we can use both Wikipedia and Forum as source domains.

5.3 Experiments

5.3.1 Dataset Description

We apply the domain adaptation technique to conduct hate speech based on three social media platforms: Wikipedia (W), Twitter (T), and Forum (F). First of all, we use the under-sample technique to get balanced datasets, Table 5.1 shows the class distributions for the three datasets we use in this chapter. Texts are encoded as 100-dimensional feature vectors of unigrams, with binary labels of hate speech and non-hate speech. For each of the three hate speech-related datasets, we pick one of them as the target domain and the rest as sources domains. For example, while applying domain adaptation based on multiple source domains, Wikipedia (W), Twitter (T) are regarded as two source domains with labeled data, and Forum is regarded as the target domain without labeled data. While applying domain adaptation based on single source domain, we only need one source domain. For example, when we regard Forum as a target domain, Wikipedia or Twitter can be regarded as a source domain, respectively. During the training phase, each source domain has (200, 400, 600, 800, 1000) labeled examples, while each target domain has (200, 400, 600, 800, 1000) unlabeled examples. We use 2000 examples from the target domain for testing. Table 5.2 shows the size of the training and test sets for our experiments in this chapter. During training, we randomly sample the same number of unlabeled target examples as the source examples in each mini-batch. Besides, we train our models in our two source domains, and test the model on the target domain directly, without using any domain information to get the "source-only" setting results. By comparing the performances of different settings with different sizes of the training datasets (200, 400, 600, 800, 1000), the effectiveness of domain adaptation technique used in our tasks has been verified.

Table 5.1: The classes distribution for three datasets

Dataset	Positive	Negative
Twitter	4000	4000
Forum	4700	4700
Wikipedia	15000	15000

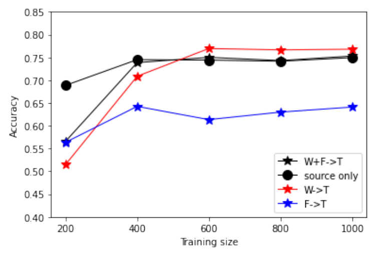
Table 5.2: The size of training and test datasets

Dataset	Training size	Test size
Twitter	200, 400, 600, 800, 1000	2000
Forum	200, 400, 600, 800, 1000	2000
Wikipedia	200, 400, 600, 800, 1000	2000

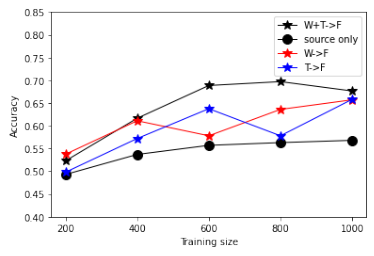
5.3.2 Pre-Processing

We remove stop words, punctuation marks, URLs, and convert all texts to lower case. We also use the lemmatization technique as the token normalization method. The details of these pre-processing steps have been introduced in Chapter 2.

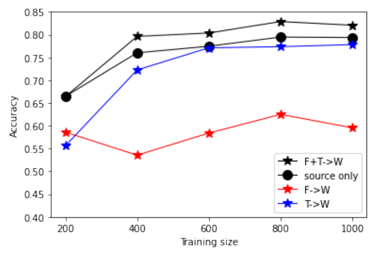
5.3.3 Experimental Results



(a) Twitter



(b) Forum



(c) Wikipedia

Fig. 5.2: Accuracy of three datasets with different training sizes

The accuracy of three datasets with different training sizes shows in Figure 5.2. First, we can observe that models using the domain adaptation technique outperform the models that are only trained on source domain without domain adaption. By comparing the single source-target pair, we can notice the different performances on our target domain task while using different source domains. (i) When the Wikipedia platform is the target domain task, the performance of using the Twitter platform as the source domain is much better than using Forum platform as the source domain. (ii) When the Twitter platform is the target domain task, the performance of using the Wikipedia platform as the source domain is much better than using Forum platform as the source domain. (iii) When the Forum platform is the target domain task, the performances between using the Wikipedia platform as the source domain and using the Forum platform as the source domain are fluctuating. This is because of the high similarity between Wikipedia and Twitter datasets and the dissimilarity between the Forum dataset to the other datasets.

Also, we can see that it is not always beneficial to naively incorporate more source domains for domain adaptation. When the target task is to conduct hate speech detection on the Twitter platform, the performance of using multiple source domains is not better than only use Wikipedia as one source domain. This is because of the high dissimilarity between the Forum dataset to the others.

5.4 Summary

In this chapter, we target the scenario of detecting hate speech on a new social media platform where no labeled data are available. To tackle this challenge, we develop a domain adversarial neural network to detect hate speech. We use the domain adaptation technique to transfer the knowledge from related social media platforms to the new platform so that the hate speech detection model trained on existing social media platforms can be adapted to detect hate speech on the new platform. The effectiveness of domain adaptation has been verified from our experimental results.

CHAPTER 6
Conclusion

With our social interactions and information being increasingly online, there are more and more cases that people express their aggressive hate, which cause varying degrees of harm to individuals or organizations. In order to make social media safe, automatically hate speech detection is an emerging task to prevent or filter hate speech. In this thesis, we propose different methods to detect hate speech in three scenarios, (i) rich resource setting, (ii) low resource setting, (iii) zero resource setting. For scenario (i), we apply traditional machine learning and deep learning methods for hate speech detection based on rich labeled data. Our experiment results show good performance of most of the existing approaches in this case. For scenario (ii), where we consider we only have limited labeled data, we develop a hate speech detection approach via fine-tuning a pre-trained language model BERT. Then we further develop a multitask learning approach to extract general information contained in our three hate speech detection tasks. We observe that the multitask learning approach can achieve better performance compared with the single-task learning approach. For scenario (iii), where no annotated data in the target social media platform is available, we propose domain adversarial neural networks to detect hate speech on the target platform. By using domain adversarial neural networks, we adapt classifiers trained on the source domain for use in the target domain and leverage annotated data from source domain to predict the hate speech on target domain.

REFERENCES

[1] L. Silva, M. Mondal, D. Correa, F. Benevenuto, and I. Weber, "Analyzing the targets of hate in online social media," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 10, no. 1, 2016.

[2] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.

[3] D. College, "Gru cell," 2020. [Online]. Available: https://sites.dartmouth.edu/odame/2020/01/

[4] S. González-Carvajal and E. C. Garrido-Merchán, "Comparing bert against traditional machine learning text classification," *arXiv preprint arXiv:2005.13012*, 2020.

[5] T. Core, "Multi-head attention." [Online]. Available: https://www.tensorflow.org/tutorials/text/transformer

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] C. Liu and K. Gryllias, "Unsupervised domain adaptation based remaining useful life prediction of rolling element bearings," in *PHM Society European Conference*, vol. 5, no. 1, 2020, pp. 10–10.

[8] E. R. Munro, "The protection of children online: a brief scoping review to identify vulnerable groups," *Childhood Wellbeing Research Centre*, 2011.

[9] M. Duggan, "Online harassment 2017," 2017.

[10] FBI, "2015 hate crime statistic." [Online]. Available: https://ucr.fbi.gov/hate-crime/2015

[11] P. Burnap and M. L. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy & internet*, vol. 7, no. 2, pp. 223–242, 2015.

[12] M. Wendling, "2015: The year that angry won the internet." [Online]. Available: https://www.bbc.com/news/blogs-trending-35111707

[13] P. Burnap and M. L. Williams, "Us and them: identifying cyber hate on twitter across multiple protected characteristics," *EPJ Data science*, vol. 5, pp. 1–15, 2016.

[14] B. Ross, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky, and M. Wojatzki, "Measuring the reliability of hate speech annotations: The case of the european refugee crisis," *arXiv preprint arXiv:1701.08118*, 2017.

[15] A. Al-Hassan and H. Al-Dossari, "Detection of hate speech in social networks: a survey on multilingual corpus," in *6th International Conference on Computer Science and Information Technology*, vol. 10, 2019.

[16] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[18] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[19] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira *et al.*, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, p. 137, 2007.

[20] C. Dictionary, "hate speech definition." [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/hate-speech

[21] Twitter, "Hate speech." [Online]. Available: https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

[22] YouTube, "Hate speech." [Online]. Available: https://support.google.com/youtube/answer/2801939?hl$=$en

[23] ILGA, "Hate speech." [Online]. Available: https://www.ilga-europe.org/what-we-do/our-advocacy-work/hate-crime-hate-speech

[24] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Eleventh international aaai conference on web and social media*, 2017.

[25] N. Tarasova, "Classification of hate tweets and their reasons using svm," 2016.

[26] Y. Chen, "Detecting offensive language in social medias for protection of adolescent online safety," Ph.D. dissertation, Penn State University, 2011.

[27] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 145–153.

[28] N. Thompson, *Anti-discriminatory practice: Equality, diversity and social justice.* Macmillan International Higher Education, 2016.

[29] Jigsaw, "Perspective api." [Online]. Available: https://www.perspectiveapi.com

[30] R. Guermazi, M. Hammami, and A. B. Hamadou, "Using a semi-automatic keyword dictionary for improving violent web site filtering," in *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System.* IEEE, 2007, pp. 337–344.

[31] A. Assiri, A. Emam, and H. Al-Dossari, "Towards enhancement of a lexicon-based approach for saudi dialect sentiment analysis," *Journal of information science*, vol. 44, no. 2, pp. 184–202, 2018.

[32] M. Wiegand, J. Ruppenhofer, A. Schmidt, and C. Greenberg, "Inducing a lexicon of abusive words–a feature-based approach," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1046–1056.

[33] N. D. Gitari, Z. Zuping, H. Damien, and J. Long, "A lexicon-based approach for hate speech detection," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 4, pp. 215–230, 2015.

[34] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–30, 2018.

[35] Y. Mehdad and J. Tetreault, "Do characters abuse more than words?" in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016, pp. 299–303.

[36] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proceedings of the 26th international conference on World Wide Web companion*, 2017, pp. 759–760.

[37] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," in *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 2015, pp. 136–140.

[38] R. Pandarachalil, S. Sendhilkumar, and G. Mahalakshmi, "Twitter sentiment analysis for large-scale data: an unsupervised approach," *Cognitive computation*, vol. 7, no. 2, pp. 254–262, 2015.

[39] T. Hua, F. Chen, L. Zhao, C.-T. Lu, and N. Ramakrishnan, "Sted: semi-supervised targeted-interest event detectionin in twitter," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1466–1469.

[40] R. Cao, R. K.-W. Lee, and T.-A. Hoang, "Deephate: Hate speech detection via multi-faceted text representations," in *12th ACM Conference on Web Science*, 2020, pp. 11–20.

[41] L. C. Yan, B. Yoshua, and H. Geoffrey, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[42] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.

[43] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[44] A. Arango, J. Pérez, and B. Poblete, "Hate speech detection is not as easy as you may think: A closer look at model validation," in *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, 2019, pp. 45–54.

[45] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate speech detection with comment embeddings," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 29–30.

[46] A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," in *Proceedings of the 10th ACM conference on web science*, 2019, pp. 105–114.

[47] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proceedings of the first workshop on abusive language online*, 2017, pp. 85–90.

[48] T. Gröndahl, L. Pajola, M. Juuti, M. Conti, and N. Asokan, "All you need is" love" evading hate speech detection," in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 2–12.

[49] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on twitter," *arXiv preprint arXiv:1706.01206*, 2017.

[50] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on twitter using a convolution-gru based deep neural network," in *European semantic web conference.* Springer, 2018, pp. 745–760.

[51] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, "Mean birds: Detecting aggression and bullying on twitter," in *Proceedings of the 2017 ACM on web science conference*, 2017, pp. 13–22.

[52] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Detection of cyberbullying incidents on the instagram social network," *arXiv preprint arXiv:1503.03909*, 2015.

[53] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[54] S. Agrawal and A. Awekar, "Deep learning for detecting cyberbullying across multiple social media platforms," in *European conference on information retrieval.* Springer, 2018, pp. 141–153.

[55] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[56] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning.* Springer, 1998, pp. 137–142.

[57] S. Indra, L. Wikarsa, and R. Turang, "Using logistic regression method to classify tweets into the selected topics," in *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 2016, pp. 385–390.

[58] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[59] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111–118.

[60] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[61] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[62] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[63] F. A. Gers and E. Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[64] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[65] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[66] CCO, "Wikipedia comments." [Online]. Available: https://creativecommons.org/licenses/by-sa/3.0/

[67] D. BALDE, "White supremacy forum data set for hate speech detection." [Online]. Available: https://github.com/djibybalde/Hate-Speech-Detection

[68] P. Mishra, H. Yannakoudakis, and E. Shutova, "Tackling online abuse: A survey of automated abuse detection methods," *arXiv preprint arXiv:1908.06024*, 2019.

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[71] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[72] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, “Multi-task sequence to sequence learning,” *arXiv preprint arXiv:1511.06114*, 2015.

[73] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, “Cross-stitch networks for multi-task learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3994–4003.

[74] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, “Facial landmark detection by deep multi-task learning,” in *European conference on computer vision.* Springer, 2014, pp. 94–108.

[75] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *arXiv preprint arXiv:1707.08114*, 2017.

[76] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[77] H. Zhao, S. Zhang, G. Wu, J. P. Costeira, J. M. Moura, and G. J. Gordon, “Multiple source domain adaptation with adversarial training of neural networks,” *arXiv preprint arXiv:1705.09684*, 2017.