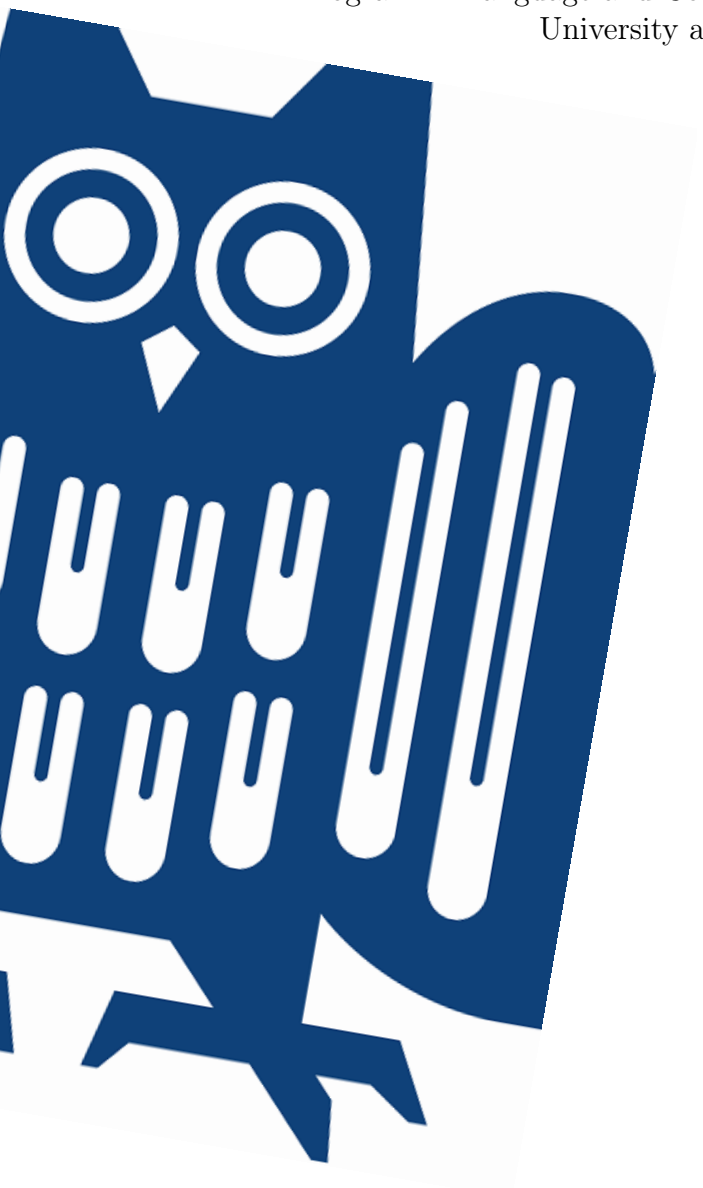
Saarland University
Department of Language Science and Technology

Master's Thesis

Exploring Features for Multi-label Hate Speech Detection

Submitted in partial fulfilment of the requirements for the degree Master of Science (MSc) in Language Science and Technology, as part of the European Masters Program in Language and Communication Technologies (LCT) at Saarland University and University of Groningen.



Md Ataur Rahman
shaoncsecu@gmail.com

Supervisors:
Prof. Dr. Dietrich Klakow
Prof. Dr. Malvina Nissim

February, 2020

Dedicated To My Loving Wife ...

Declaration of Authorship

I, Md Ataur Rahman, hereby confirm that this thesis titled, 'Exploring Features for Multi-label Hate Speech Detection' and the work presented in it are my own. I confirm that:

- This work was done wholly while I was a candidate for the degree of Master of Science(MSc) in Language Science and Technology, as part of the European Masters Program in Language and Communication Technologies (LCT) at Saarland University and University of Groningen.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

I also assure that the electronic version that I have submitted is identical in content to the printed version of the Master's thesis.

Signed: 

Date and Place: 28/02/2020, Saarbrücken

Acknowledgements

First and foremost, I want to express my unswerving gratitude to the Almighty Allah for his immeasurable kindness and blessings for retaining my mental and physical fitness to finish this sophisticated work.

This research would not have gotten its structure without the general support and aid from the Erasmus+ program of the European Union which offered me with the opportunity for my Master's studies through scholarship. The completion of this dissertation is not a consequence of my individual effort but is an aggregate of the combined effort of many people.

I want to express my sincerest gratitude to my supervisor, Prof. Dr. Dietrich Klakow, who is one kind of turning point in my life. His broad expertise in Machine Learning and Linguistics and interest to spread that wisdom and knowledge are really praiseworthy. I honestly admire his aptitude of understanding and endurance in dealing with students and wish to have such kind of qualities in me. I do not hesitate to accredit him as one of the most exceptional academic scholars I have encountered so far.

I would like to convey my esteemed indebtedness to my second supervisor, Prof. Dr. Malvina Nissim from the University of Groningen for her guidance, patience and support, specially for encouraging my research by allowing me to pursue my own interests. I consider myself very fortunate for being able to work with a very considerate and supportive person like her.

I also would like to thank Dr. Gosse Bouma for supporting me whenever I encounter some difficulty; his constant supports, motivation, guidance, and advice will always hold a strong influence in my life. Special thanks to Ms. Bobbye Pernice and all the other coordinators giving me suggestions and directions directly or indirectly throughout various stages of my studies.

I also want to recollect all of my classmates who always inspired, helped and motivated me. Special thank goes to Morgan Wixted and Xinia for their support during the submission of my thesis. I also wish to thank Ms Dana Ruiter for her continuous support during my thesis work.

Finally, I would like to show my gratitude to all of my teachers who helped me a lot during my Master's program, without their teaching, it would have been impossible for me to learn even a bit of what I have learned all these years.

"We have to bear in mind that words can kill, words kill as bullets do."

- Adama Dieng

Abstract

The correlation between opinions and text has been an enchanting topic for centuries. Today the web has become an excellent means of expressing opinions about anything, especially with the increasing popularity of social media. Such social platform seems to be a fertile ground for hate speech and hateful comments. Hate speech is commonly defined as any communication that disparages a person or a group based on some characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics [1]. In simple terms, any communication that may hurt the feelings of a particular group or person is termed as hate speech. Due to the massive size of online contents that is being produced on a daily basis, it is quite impossible to manage and separate all these hateful allusions from good contents. This far-fetched problem of manually managing online hate speech annexed the need for automated detection and classification of hateful activities. As automated computational approaches based on machine learning (ML) techniques became a popular means of tackling these hateful speeches, feature selection plays an important role in terms of the performance of the classification. In this thesis, we intend to explore the different combination of features, including state-of-the-art input representation such as Byte Pair Encoding (BPE) and word/document embeddings with classical ML algorithms such as Support Vector Machines (SVM) for the classification of hate speech from text. The incentive behind selecting classical ML came from the fact that given the size and nature of different hate-speech corpora, whether or not it is possible to compete with state-of-the-art models such as BERT [2] if we incorporate advanced features.

Contents

Abstract

Contents

Chapter 1

INTRODUCTION

`Hate` - a mental state in which someone tries to humiliate a person or group by rejecting or attacking their faith or other identity factors. It can be seen as a mutation of the seven deadly sins [3], namely pride, greed, lust, envy, gluttony, wrath and sloth. Throughout history, it is evident that almost all hate crimes are preceded by hate speech. The genocide against the Tutsi in Rwanda started with hate speech. The holocaust did not derive from the gas chambers, it began long before with hate speech. What we have seen in Myanmar against the Rohingya population, also sprang from hate speech. And today what we are witnessing around the world, when we see the way migrants and refugees are being vilified, is because of the rise of extremists everywhere. We need, therefore, to make every effort to address this hate speech.

1.1 Motivation and Research Objectives

For ages, social platforms such as YouTube, Facebook and Twitter have been struggling with the problem related to hateful posts and comments. Nearly hundreds of millions of money and a countless number of human resource are being invested each year to tackle this problem [4]. Still, they are failing to crack it down mainly because such measures mostly involves human intervention and manual review of the online contents to distinguish and to remove the offensive materials. This process is time-consuming, labour intensive and not scalable or feasible in reality [5][6].

One of the critical hurdles of automatic detection of hate speech from social media data that comes in the form of text is to define the border that separates hateful contents from other instances of language. A detection methods that rely only on the lexical features, tend to classify all such documents containing some particular terms or keywords as hateful text. As a result, those systems typically have low precision. The urge for an automated, scalable scheme of detecting hate speech has captivated an escalating number of researchers from both the Machine Learning (ML) and Natural Language Processing (NLP) domains in the recent years. State of the art systems that try to solve this problem mainly categorize it as a supervised text classification task [7]. It is thus typically tackled using two categories of algorithms: *classical machine learning* and more advanced neural networks that incorporate *deep learning* paradigm. The former approach heavily relies on hand-crafted feature engineering, which is then

fed to a classifier such as Logistic Regression (LR), Naive Bayes (NB) or Support Vector Machine (SVM). Because of the lack of appropriate datasets and computational complexity, relevant research has shown that traditional Machine Learning methods perform significantly better than their neural network counterparts in the hate speech domain if we tweak them with advanced feature engineering techniques.

Thus, the main incentives behind this thesis was to investigate whether or not it is possible to beat state-of-the-art neural based methods using more conventional feature based classifiers such as SVM's in the realm of textual hate speech classification. We explored many preprocesing and features selection techniques in this thesis. We investigated both traditional features such as char n-grams, Tf-Idf etc., and more advanced features such as byte pair encoding, sentence/document embeddings on various settings in both binary and multi-level classification task for three different languages (English, German and Hindi). By employing a combination of Byte Pair Encoding with document-level embeddings we found out that our system was able to beat almost all the top performing models (including our own BERT model [8]) in the *Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC) shared task 2019* [9].

Thus the main objectives of this thesis was:

- To explore different combination of features with classical ML techniques (e.g., SVM) for hate speech detection from text.
- To check the performance of such classifiers using unconventional input representation techniques such as byte-pair/word/document embeddings and compare them with state-of-the-art neural models such as BERT.
- To undertake an additional investigation to understand the impact of external data and feature engineering in classic methods in the domain of multilingual hate speech.

1.2 Structure of the Thesis

This remaining of this thesis will be structured as follows:

Chapter 2 introduces some of the background knowledge and related work:

- Definitions and related terminologies.
- An overview of the application fields of hate speech detection.
- Some of the most recent approaches to hate speech detection.
- And finally this chapter ends with a discussion of the challenges of detecting hate speech.

In chapter 3, we present several hate speech corpora that was being used during our experimentation.

Chapter 4 presents an overall interpretation of the methodology and design of our hate speech models including a discussion of the specific tasks that we have solved.

In the 5th chapter, we've discussed about the results and evaluation of our system. Starting with a detail discussion about the evaluation data sets, followed by a description of the evaluation strategy, and finally we've conclude this chapter by manifesting a comparative performance with state-of-the-art models.

Finally, we conclude this thesis with a discussion of the implication of our system and an overall epilogue in Chapter 6.

Chapter 2

BACKGROUND AND RELATED RESEARCH

This chapter will provide a summary of the work conducted so far on hate speech detection. We address the topic systematically, providing both theoretical and practical aspects and giving an overview of the most recent approaches. While doing so, we will focus on several shared tasks in our literature survey that are aiming at improving the field constantly. The first section concentrates on defining hate speech and understanding the hate speech related terminologies with a follow-up discussions on the importance of this research and application fields. Finally, we will try to throw a light on the possible challenges that one might encounter when investigating the topic.

2.1 The Backdrop

2.1.1 What is Hate Speech?

Different organizations and scholars have defined hate speech based on different perspectives over the year. Some of the most influential definitions of hate speech are conferred below:

- According to the definition of the Council of Europe's Committee of Ministers in 2005 - "*Hate speech* shall be understood as covering all forms of expression which spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance, including intolerance expressed by aggressive nationalism and ethnocentrism, discrimination and hostility against minorities, migrants and people of immigrant origin". In this sense, hate speech covers comments which are necessarily directed against a person or a particular group of people.

- In 2010, ILGA-Europe defined *hate speech* as "the public expressions which spread, incite, promote or justify hatred, discrimination or hostility towards a specific group. They contribute to a general climate of intolerance which in turn makes attacks more probable against those given groups".

- Facebook defines *hate speech* as a "directed attack on people based on what we call protected characteristics - race, ethnicity, religious affiliation, sexual orientation, sex, gender, gender identity and serious disability or disease".

- Twitter's guideline says "Users may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease. We also do not allow accounts whose primary purpose is inciting harm towards others on the basis of these categories. The consequences for violating the Twitter rules vary depending on the severity of the violation. The sanctions span from asking someone to remove the offending Tweet before they can Tweet again to suspending an account

- According to YouTube's policy - "We encourage free speech and try to defend your right to express unpopular points of view, but we don't permit hate speech. Hate speech refers to content that promotes violence against or has the primary purpose of inciting hatred against individuals or groups based on specific attributes, such as race or ethnic origin, religion, disability, gender, age, veteran status, sexual orientation/gender identity. 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 if the primary purpose of the content is to incite hatred against a group of people solely based on their ethnicity, or if the content promotes violence based on any of these core attributes, like religion, it violates our policy.

- The most widely accepted definition of hate speech was given by Davidson et al. [10], which defines it as "speech that targets disadvantaged social groups in a manner that is potentially harmful to them".

2.1.2 Related Terminologies

Before diving deep into the problem of classifying hate speech, it is important to know about hate speech related terminologies and its sub-types.

- `HATE` - is an emotion that can invoke feelings of animosity, anger, or resentment, which can be directed against certain individuals, groups, entities, objects, behaviors, concepts, or ideas [11]. In an abstract way, all the subcategories of hate speech can also be termed as hete and thus making it a general expression of hatred.

- `OFFENSIVE` - speech or writings which are degrading, dehumanizing, insulting an individual, threatening with violent acts [9].

- `PROFANITY` - usage of profane/swear words such as ass, pissed-off, fuck, etc. However, this most likely does not have any particular target [9].

- `INSULT` - act or speech that clearly wants to offend someone. For example '*you are fake*'.

- `ABUSE` - unlike INSULT, this does not just insult a person but represents the stronger form of abusive language (e.g., *fucking retard*).
- `TOXIC` - some form of rude, disrespectful comment that is likely to make someone leave a discussion (e.g., *Atheism is full of bias shit*).
- `THREAT / AGGRESSION` - is an action or response by an individual that delivers something unpleasant or the intention to harm another person [12][13]. It might be overt or covert, often harmful, social interaction with the intention of inflicting damage or other unpleasantness upon another individual and may occur either reactively or without provocation [14].
- `IDENTITY HATE` - is a homophobic or transphobic hate incident if the victim thinks it was carried out because of hostility or prejudice based on sexual orientation or transgender identity [15].
- `Sexism` - any act, attitude, or institutional configuration that systematically subordinates or devalues women. Built upon the belief that men and women are constitutionally different, sexism takes these differences as indications that men are inherently superior to women, which then is used to justify the nearly universal dominance of men in social and familial relationships, as well as politics, religion, language, law, and economics [16]. One such example of sexiest comment could be - *'She swims like a man'*.

2.2 Applications of Hate Speech Detection

Although hate speech is not a new thing but with the dominance of social media usages such as in Facebook, Twitter and YouTube, the problem is becoming more prominent and visible as ever. Now more and more people are being exposed to hate crimes, and social media platforms are acting as an ignition to these hateful activities. Thus the impact of accurately detecting such offensive speech is enormous, and numerous application areas will be profoundly influenced. Some of the implications are listed below:

- Hate speech is undoubtedly a common phenomenon on the Internet [17][18] and in extreme cases, it can be the cause of individuals being severely harmed. Thus the significance of identifying and managing hate speech is apparent from the obvious association between hate speech and actual hate crimes.
- The classification of posts or tweets in social media platforms into specific type of hate expressed (e.g., homophobic, abusive, xenophobic etc.) could be utilised as a source of data by legal organizations to tackle this problem.
- Such systems could be adapted in other domains such as police investigation, gender violence, counter-terrorism, or for cyberbullying prevention, and thus extending knowledge and capacities of such organisations to identify possible criminal content.

- Prediction and advanced warning systems could allows us to take early action against the possible impact of hateful crimes that might happen due to the original hate speech.

- Automatic removal of toxic content from social media platform is a direct application as it will ensure safeguarding the integrity of moderate users.

- Given the alarming outgrowths that cyber aggression has on the victims, and the accelerated extent among the users of the internet, especially within youngsters, it is imperative to realise how it occurs to halt this from escalating. This has great significance on the apprehension of cybercrime, cyberextremism and hateful propaganda.

2.3 Literatures

The literature reviews in this section is arranged in a way that reflects both feature variations and modelling techniques.

2.3.1 Lexicon Based Approaches and Shallow Features

Lexical or dictionary-based methods are the most fundamental strategies in natural language processing. These techniques mostly use a table of lexicon such as WordNet [19] to find out certain words or its meaning representations to be used as features. One such approach is to store a typical list of abusive words that be can be used for blacklisting specific offensive terms or to classify them further [20]. Apart from blacklisting, Gitari et al. compiled a collection of hate verbs that represents violent actions or behaviour in an endeavour to construct a resource that enables lexical methods to be more accurate and applicable in hate speech domain [20].

Shallow or surface features also played a pivotal role in the improvement of earlier systems and are still useful today. These features can also be used in addition to lexical sources to mitigate their dependency to a specific knowledge base and to improve the accuracy. Surface features typically include elements such as punctuation symbols, document length, emoticons, capitalisation or foreign words and numerals [7]. These are usually extracted manually during the pre-processing step.

A study done by Davidson et al. provided more insights into the role of certain words with variations of spelling in determining the categories and extent of hate speech as features [10]. The authors also suggested adding sentiment scores aid in achieving better accuracy while distinguishing abusive language from other groups of hate.

One of the biggest deficiency of lexicon-based methods is the bag of word assumption. It considers each idea or concept through a single word without acknowledging contextual information or semantics. But words in any languages can have more than one meaning depending on the context and thus could be ambiguous. This also leads to the problem of dealing with a very large vocabulary, as denotational representation normally have a localist meaning which leads to a very large embedding vector [21]. Another dilemma of lexical scheme is that it fails to capture the implicit forms of hate

that do not contain any profane or abusive words. It is also hard to capture permutations of characters in different swear words that are in euphemistic spelling form (e.g., b*tch, bth) [22].

2.3.2 Language Models: Word and Character n-grams

Language models oppose the concept of denotational representation (lexical approach) by asserting that individual word in a document often fails to imply intended meaning without its context; that is why lexical methods are not very powerful. In other words, this is saying that "*you shall know a word by the company it keeps*" [23]. This idea was later expanded on by Zellig Harris in 1954, who stated that "the restrictions on relative occurrence of each element are described most simply by a network of interrelated statements, certain of them being put in terms of the results of certain others" [24]. Interestingly, this distributional approach to quantifying word meanings is still used in most techniques today.

A simple yet elegant example of this distributional approach is `n-gram` methods. In the context of computational linguistics, an n-gram is a contiguous sequence of n items, where the item in question is usually a `character` or a `word`. For example, the character tri-grams of the name *banana* would be *ban, ana, nan, ana*. By collecting a large number of n-grams over a large corpus, the n-grams can act as a probabilistic language model that predicts the next item (i.e. a character or a word) in a sequence.

Greevy et al. [25] used SVM's with simple features like bag of words and `bi-grams` to categorize *racist* web pages. With a small dataset of around 1k documents they achieved 92.78% precision score using a polynomial kernel.

Perhaps one of the most common dataset for English hate speech is the Waseem and Hovy corpus [6]. The corpus is around 17k based on *sexism*, *racism* and *none*. They have also used a Logistic Regression (LR) model and considered the influence of various features such as `word/character n-grams`, gender, location and length to evaluate their model on the dataset. Character based n-gram (n = 1 to 4) along with gender feature produced an average f1 score of 0.7393 on a 10-fold cross validation using the LR classifier.

Applied to hate speech, Mehdad et al. also found that character n-grams outperform state-of-the-art lexical methods as well as word n-gram methods [22]. However, n-gram models also have their shortcomings. For instance, in bag-of-n-grams, the word order is not preserved, implying that different sentences can have precisely the same representation, as long as the same words are being used. For example, the sentences "*I want tea instead of coffee*" and "*I want coffee instead of ice tea*" would result in the same representation, although they are opposing statements. Moreover, bag-of-word models would pool ambiguous words together, such as "break", which can be seen as a noun or a verb. Badjatiya et al's comparative study showed that n-grams performed the worst out of all tested embeddings methods [26].

2.3.3 Word Embedding Techniques

The most widespread embedding method employed in natural language processing is the word embeddings (such as `Word2Vec`) [27]. It can be seen as a shallow two-layer

Neural Network, trained to reconstruct linguistic contexts of words. This in terms produces a more sophisticated version of a feature vector where semantically related words may end up having similar vector representations relative to each other (e.g. 'earth' and 'world' will most likely to have a similar word vector). Therefore, capturing the relative semantics of a word, similar to Harris' idea of the "network of interrelated statements" [24]. The word vectors can then be used as a classification feature and have been shown to outperform bag-of-words approaches for a number of general NLP learning tasks [7] as well as hate speech specific classification tasks [26].

Word vectors can either be trained directly using the training data or can be obtained from pre-trained models such as FastText [28] which provides already trained word vectors from a large text corpus. The main advantage of using FastText over self-trained vectors is the inclusion of character n-grams, which allows computing word representations for words that did not appear in the training data. FastText has proven to generate better word embeddings for rare words or out-of-vocabulary (OOV) words. FastText outperformed pre-trained Word2Vec models on a larger text corpus (e.g. Google News, Wikipedia Corpus) [29] [28].

A slightly more effective extension of word2vec is Global Vectors (GloVe) introduced by Pennington et al. [30]. Global vectors are computationally less expensive as compared to word2vec. While word2vec captures the co-occurrence of words one window at a time, GloVe counts the co-occurrences of two words within the entire text corpus in a large co-occurrence matrix and then minimizes the square distance between the dot product center and context vectors and the log product of the co-occurrences [30]. On top of its improved computational efficiency, GloVe has been proven to outperform Word2Vec and FastText on a range of learning tasks [30] and came out top in Badjatiya et al.'s comparative hate speech study [26]. However, the performance of GloVe is similar to that of other word embedding techniques which are heavily dependent on the training corpus.

2.3.4 Sentence and Document Embeddings

Pre-trained embeddings have become a core trend in Natural Language Processing (NLP), largely due to its capabilities for representing complex semantic characteristics that natural language possesses. Whether in their guise as a representation that it holds for the neural network to work with or simply as the agglutinating property of language units that can be described by similar units in a certain way, it is truly one of the major breakthroughs in the computational representation of languages.

For years word embedding methods such as `word2vec` [27] or `GloVe` [30] are the most popular means of text representation and seems to be the only de facto when it comes to working with neural nets. While words are good way to represent meanings in some extent, it often fails to show us the big picture in contrast to sentences. There are two significant intricacies of applying word vectors for hate speech classification. Firstly, hate comments require to be classified as sentences and documents instead of merely taking the sum of individual tokens or words. Furthermore, the length of words in each comment differs, which is problematic while using neural networks, as they require a fixed-size input [31]. A simple way to resolve this problem is to either take the word centroid, which adds all the word vectors of words in the sentence

together, or averaging the word vectors. From this, we would be able to compute the similarity between two sentences by looking at its centroid distance. This technique of `sentence embedding` is not new and people have been trying several other methods such as mapping word vectors to sentence vectors through the use of more conventional architectures like RNN and CNN variants [31–35] or through the use of unsupervised approach like SkipThought vectors [36] to construct embeddings from sentences.

A commonly practised approach for combining word vectors within sentences is to apply CNN on top of the word vectors such as in [34]. The author suggested padding the vectors in order to convert them into an equal dimension, then mapping the words in the new sentences into word embeddings, employing a convolutional layer followed by a pooling and a fully-connected layer to get the final sentence representation. The paper confirms that this procedure achieves better results by utilizing word2vec even without any tuning or random initialization and also proved to be quite useful in both Zhang's [29] and Badyathia's [26] investigation of deep learning approaches.

However, with these methods, one runs into the same problem as with n-gram methods, i.e. corresponding vectors for sentences containing the same words but having contradicting meanings. Moreover, averaging word vectors for sentence representations are found not to be very efficient [22]. This is also the reason we restrained ourselves from using averaged word vectors.

Doc2Vec

Although few more recent work on sentence embeddings such as Google's *"Universal Sentence Encoder"*[1] [37] that uses transformer-based architecture and Facebook's *"Universal Sentence Representations"*[2] [38] shows promising results on some transfer learning tasks (e.g., sentiment analysis, semantic textual similarity), they are however not generalized, and it takes a lot of computing power in order to train them from the sketch.

Currently, Doc2Vec [31] is the state-of-the-art technique for comment or sentence embedding, which proposes to train a paragraph vector as an unsupervised learning algorithm that learns fixed-dimensional distributed feature representations for variable-length pieces of texts. Mikolov and Le have shown that this technique achieves better results than any other embedding technique on a sentiment classification task [31]. In addition to this, Doc2Vec was found to be the most effective technique for a binary hate speech classification problem from Yahoo Research [39]. This is the primary reason for which we adhere to using Doc2Vec (using `gensim` library [40]) in our research.

2.3.5 Classical Vs. Deep Learning Techniques in Recent Shared Tasks

This section will start briefly with a more generalized discussion about a few of the earlier work on hate speech detection that compares classical machine learning with deep learning methods. We will try to conclude our literature survey by throwing

[1] `https://tfhub.dev/google/universal-sentence-encoder/4`
[2] `https://github.com/facebookresearch/InferSent`

light on the more recent shared tasks on hate speech detection since our work mainly revolves around the recent HASOC [9] shared task. This, in essence, will also consolidate the importance of classical machine learning approaches in tackling hateful contents.

A comparison between SVM, Seq2Seq and FastText classifier was shown in one of the paper [41] by Akshita et al.. They build a corpus on top of Waseem and Hovy dataset which incorporated *benevolent*, *hostile* and *others* to categorize hateful activities. Even with only tf-idf (term frequency–inverse document frequency) feature, SVM performed quite well on average compared to the seq2seq model. For class benevolent and others SVM scored highest among all the classifiers.

In a paper [10], the authors fond that *racist* and *homophobic* tweets are more likely to be classified as Hate Speech. while *offensive* (but not hate speech) tweets are those that are are generally classified as *sexist* tweets. This paper can be considered as one of the most influential paper that combines almost all the popular classical approaches to machine learning with a big set of feature selection techniques. They used lowercasing, stemming, n-gram(1, 3), tf-idf, parts of speech (POS), sentiment lexicon score and char/word/syllables length features. With an f1-score of 0.90, Logistic Regression classifier showed the best result among the other classifiers (Naïve Bayes(NB), Decision Tree (DT), Random Forests (RF) and SVM).

Perhaps the best comparison between classical machine learning technique with state-of-the art deep neural network model was done by Zhang et al. [42]. They have used seven publicly available English datasets on a CNN and Long short-term memory (LSTM) based model to compare the outputs with SVM's that uses feature engineering. As features they have used fairly simple set such as Tf-Idf, word n-gram (1, 3), mentions, hashtags, lengths (char and word), POS-ngrams (1, 3), Sentiment features etc. On the biggest corpus (WZ-L), SVM outperformed the DNN based model. SVM with feature selection scored an f1 of 81.4 the CNN and LSTM based model scored 80.2 respectively.

On a dataset of unshared task for the Workshop on Abusive Language Online (ALW1), Ji Ho et al. used Logistic Regression, SVM and Hybrid Convolutional Neural Network(CNN) models consisting of wordCNN and CharCNN [43]. They used one step and two step methods where in one step method three way multi-class classification was done. In contrast, the two step classifier consisted of two binary classification task between *none* vs *abusive* and a subsequent *racism* vs *sexism* (if abusive). Although, LR classifier in two step classification showed the best result (f1-score of 0.824) but SVM on the single *racism* vs *sexism* task gave unparalleled performance scoring 0.952 points on f1.

In one of the recent GermEval 2018 Shared Task [44], a total of 20 teams participated to tackle the problem of offensive language identification from German tweets. The task was divided into two categories (binary vs multi-class) and the best teams from each of the discipline scored 79.53 (coarse grained) and 73.67 (fine grained) in terms of accuracy measure. One of the teams from Saarland University [45] participated on the Binary task of detecting offensive (or non-offensive) language. They tried three different methods namely – Lexicon based, SVM and Neural Network, and tried to compare the results for these classifiers on both monolingual and crosslingual settings. As lexicon, they semi-automatically created a total of 1566 unigrams by translating English abusive words and also manually adding German abusive words from Wik-

tionary. They reported a best SVM model based on character n-grams(n=6), word embeddings using COW16 [46] and the task-specific lexicon as features which acquired an F1 macro score of 77.4. The team also experimented with two versions of neural nets (LSTM and GRU) and the GRU achieved 73.5 in terms of F1 macro score. Their best submission containing an ensemble of all the 3 best classifiers gained an accuracy of 75.62 and F1 (macro) of 72.05 on the final task.

Another team [47] used a combination of CNN and RNN (with GRU) to tackle the same problem. For their experiment with different combinations of features, they have used 10-fold cross validation techniques over the training data. They also separated the final 500 examples from the training set to finally evaluate the best versions of each of their models. They claimed that bidirectional gated recurrent units (GRU) performs better than LSTM in the case of RNN because of small number of training examples and also lowercasing the words seems to give higher results. Among three runs, the best ensemble produced an accuracy of 77.27% and a macro F1 of 74.64.

In another multi-lingual shared task named TRAC (2018) [48], a total of 30 teams submitted their test runs to classify *overtly aggressive*, *covertly aggressive*, and *non-aggressive* texts. The best system obtained a weighted F-score of 0.64 for both Hindi and English on the Facebook test sets, while the best scores on the surprise set were 0.60 and 0.50 for English and Hindi respectively.

In a recent shared task named "Hate Speech and Offensive Content Identification in Indo-European Languages" (HASOC) [9], our team LSV-UdS [8] participated on *Task-A* (binary classification task - `hate` vs. `non-hate`) and *Task-B* (multi-class classification task - `profane/offensive/hate/none`) for all three of the languages (`Hindi`, `German` and `English`). During the competition, we employed both BERT and SVM based models and explored the possibility of data augmentation by incorporating and mapping other hate speech corpora. Note that during the shared task, the SVM classifier that we employed, only used more traditional features such as *tf-idf*, *word/byte-pair/char level n-grams* etc. Even with that, we got quite a surprising result that was able to compete with our BERT based model in several categories. We secured 1^{st} position for German Task-B and 2^{nd} position for German Task-A. Our Hindi model was also able to achieve 2^{nd} position for the binary task. Chapter 5 delineates a brief about our results.

From this point on, during the original thesis work, we tried to go beyond just using conventional features. With more unconventional features and input representations that are quite popular with deep learning methods, we were able to beat most of the ranked teams by a significant margin. We achieved better results than our own state-of-the-art BERT models by using a vanilla SVM classifier. To best of our knowledge, this is the first such research effort that combines Doc2Vec and Byte Pair Encoding (BPE) along with SVM to tackle hate speech task.

2.4 Challenges of Hate Speech Detection

This section will highlight various aspects which makes the task of automatic hate speech classification tricky. Firstly, we will focus on the fact that there is a great dispute among the most traditional definitions of the term hate speech and its subcategories.

Then we will discuss about the limitations regarding the interchangeability of corpus for different hate speech task.

- The term hate is vague by nature, that is one sentence that may look like hateful to a person or in one culture could be treated as sarcasm in another part of the world. Also, detection of offensive contents is not as simple as keyword look-up problem. It depends on the context and in the inherent nature of the topic being dealt with.

- Hate speech is a fore-and-aft phenomenon that evolves in parallel with language evolution. Its apprehension can be difficult when it comes to classifying the language being used by the youngsters as they use a metaphorical way of communication [49]. Thus, the socio-linguistic attribute of this phenomena in different social media is of significant interests for researchers [50].

- The complexity of annotating hate speech into different sub-classes are heavily biased towards the cultural, social, political and religious belief of the annotators. For example, if someone is annotating something related to socio-politics, (s)he needs to have proper background knowledge about political parties and voters in that particular region where the text originated.

- Annotators agreement in most of the cases are observed to be very low [51], meaning that the same task would probably be quite difficult for the machine learning systems. For instance, if we want to annotate tweets about migrants, we might need to include real immigrants in order to realize their perspective. The same thing which might look like terrorism to one country could also be seen as resistance to the forced occupation.

- Most of the offensive languages in social media such as tweeter are spelled in informal or with ungrammatical forms. People even use asterisk or other symbols instead of characters inside profane words (e.g., f**k, d@#$k) which is unpredictable and challenging to interpret for the algorithms.

- Finally, there is considerable confusion even within the sub-classes of hate, such as `abusive` vs. `insult` and so on. A clear distinction is often impossible in such scenarios. Thus, although a prediction is incorrect when compared to the gold label, it might actually not the case.

Chapter 3

DATASETS

Hate speech is a highly sophisticated phenomenon which is heavily dependent on the geo-political background and the data being analyzed. Moreover, there is no clear standardization when it comes to the subcategorization of hateful speech, and almost all the available datasets differ in the way they have looked into the problem and categorized them. Thus each of the datasets has its own version of class labels, and it is difficult to use external data that really helps. Nevertheless, we have used several external corpora apart from the primary HASOC evaluation corpus in our experiments which we are going to discuss in this chapter.

3.1 External Corpora

Almost all the datasets in hate speech domain are for English, and we could find only a handful of gold standard corpus for other languages (German and Hindi). The availability of datasets for other languages apart from English is another motivation to use classical ML approaches. We have used the following external datasets in our experiments (Table 3.1):

Name	***Lang.***	***Source***	***Size***	***Categories***
Founta	en	Twitter	100 k	abusive, hateful, normal, spam
Davidson	en	Twitter	27 k	hate, offensive, other
Waseem	en	Twitter	24 k	racism, sexism, other
SemEval	en, es	Twitter	13+6.6 k	hateful/not, aggressive/not
Kaggle	en	Wikipedia	300+ k	toxic, severe-toxic, obscene, threat, insult, identity-hate
TRAC	en, hi	Facebook	15 k	overtly aggressive, covertly aggressive, non-aggressive
GermEval	de	Twitter	5 k	profanity, insult, abuse, other

Table 3.1: External Hate Speech Datasets for Different Languages.

We will briefly describe each of the above datasets below.

3.1.1 Founta Corpus

The *Founta* (FO) [52] corpus[1] is based on around 100k English twitter data collected using boosted random sampling techniques. It focuses on `spam`, `hateful` and `abusive` speech. The annotation was mainly done via crowdsourcing.

3.1.2 Davidson Corpus

The *Davidson* (DA) [53] corpus[2] is also known as the *Cornell University* hate corpus, is a collection of around 27k tweets. The corpus was collected based on crowd-sourced offensive keywords. Three or more *Figure Eight*[3] (formally known as CrowdFlower) users coded the tweets into either `offensive`, `hate speech` or `other`.

3.1.3 Waseem and Hovy Corpus

With a total of 24k tweets, the *Waseem and Hovy* (WA) [54] corpus[4] tried to distinguish between `racist` and `sexist` tweets. The authors employed a keyword search method based on slur to collect the data, and the inter-annotator agreement was $K = 0.84$. They also produced a list of most indicative features (character n-grams) from their dataset.

3.1.4 SemEval 2019 - Task 5 Corpus

The SemEval 2019 - Task 5[5] (SE) also known as *hatEval*[6] shared task [55] was mainly concerned about `hateful` and `aggressive` content against women and immigrants in twitter. There was both Spanish (13k) and English (6.6k) annotated tweets where around 9k data contained hateful speech regarding immigrants, and nearly 10.5k tweets were against women. We only used the English dataset in our research. They also used the *Figure Eight* platform for the annotation.

3.1.5 Kaggle Corpus

The *Kaggle* (KA) [56] corpus[7] is a huge collection of Wikipedia comments that was mainly used for *Toxic Comment Classification Challenge* by Kaggle during 2018. It includes more than 300k hate-related contents which belongs to classes ranging from `toxic, severe toxic and obscene` to `threat, insult` and `identity hate`. The dataset was also collected using a boosted random sampling method. One downside of this corpora is that around 60% of the data belongs to *none* label.

[1] `https://github.com/ENCASEH2020/hatespeech-twitter`
[2] `https://github.com/t-davidson/hate-speech-and-offensive-language`
[3] `https://www.figure-eight.com/`
[4] `https://github.com/zeerakw/hatespeech`
[5] `http://alt.qcri.org/semeval2019/index.php?id=tasks`
[6] `https://competitions.codalab.org/competitions/19935`
[7] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data`

3.1.6 TRAC - 1 Corpus

We used the *TRAC-1* 2018 (TR) shared task [48] corpus[8] that focuses on the `overtly`, `covertly` and `non-aggressive` Facebook comments. This dataset contained around 15k data from both `Hindi` and `English`. This dataset is a subset of another larger corpora developed by Kumar et al. [57]. There are however, two major issues regarding this dataset. Firstly, the English dataset contains code-mixed English-Hindi data as well. The second problem is regarding the annotations, as some of the examples contains implausible class labels. Nevertheless, this dataset gave us a huge boost in the results when applied with HASOC dataset mostly due to the nature of the data (as both the corpora came from the Indian subcontinent).

3.1.7 GermEval Corpus

In order to augment the *German* data, we opt the GermEval 2018 (GE) [44] corpus[9]. The data in this corpus mainly has four categories namely - `profanity`, `insult`, `abuse` and `other` which aggregates to around 5k tweets.

3.2 HASOC Dataset

Since this is the primary evaluation dataset for this thesis, we will have an elaborate discussion on the data. We mainly worked with both **binary** and **multi-lebel** categories of data from three of the languages (Hindi, German and English). The binary class contains `hate and offensive` (HOF) vs. `non-hate` (NOT) while the multi-label data further divides the HOF into more fine-grained categories namely - `hate` (HATE), `offensive` (OFFN) and `profanity` (PRFN). The definitions of each of the above categories can be found in section 2.1.2. For all the three languages, the dataset was sampled partially from Facebook and mostly from Twitter following a systematic guideline. According to the original paper [9] -

> "At first, the organisers sought for heuristics that typically describes hate speech in popular online discussion forums. They pinpointed relevant topics toward which various hate content might be generated. Based on those topics, keywords and different hashtags were collected and utilised in order to extract the tweets for all three languages. For randomly found tweets, the id of some of the authors were also collected in order to get the entire timelines tweet. To increase the variety, they also crawled the previous posts of every user they collected the tweets from. Thus making the system less susceptible to bias. This technique was inspired by the GermEval shared task [44]."

In a nutshell, the tweets were extracted through the use of keywords and hashtags, which included hateful content. Below (Table 3.2) the statistics of the HASOC training and test set is given based on the class distribution.

[8] `https://sites.google.com/view/trac1/shared-task`
[9] `https://github.com/uds-lsv/GermEval-2018-Data`

Data	Lang.	NOT	HOF	HATE	OFFN	PRFN	Total
Training	en	3591	2261	1143	667	451	5852
	de	3412	407	111	210	86	3819
	hi	2196	2469	556	676	1237	4665
Test	en	865	288	124	71	93	1153
	de	714	136	41	77	18	850
	hi	713	605	190	197	218	1318

Table 3.2: Class Distribution for Official HASOC Training and Test Set.

3.3 Mapping Between Corpora

In our investigation, we have experimented with several external hate-speech corpora (Table 3.1) and their impact on the classification performance. A mapping between comparable classes was done in such cases, which are delineated in table 3.3 along with the task-specific distribution of classes for each of the languages.

Corpora	Source	Task A	Task B	Mappings (A)	Mappings (B)
Kaggle	en	143.3/16.2	0.3/14.5/1.4	$\forall c = 0$ →NOT $\exists c = 1$ →HOF	obsc →PRFN id.hate →HATE rest →NONE
Davidson	en	2.5/12.3	0/11.5/0.8	none →NOT hate, off →HOF	offn →OFFN hate →HATE
Founta	en	53.9/32.1	0/27.2/5.0	normal →NOT hateful, abusive →HOF	abusive →OFFN hateful →HATE
TRAC	en hi	7.4/9.8 3.4/13.8		none →NOT aggr →HOF none →NOT aggr →HOF	
GermEval	de	3.3/1.7	0.1/0.6/1.0	none →NOT hate →HOF	prfn →PRFN ins →OFFN hate →HATE
SemEval	en	5.8/4.2		not →NOT hate, aggr →HOF	
Waseem	en	11.1/5.1		other →NOT racism, sexism →HOF	

Table 3.3: The mappings between classes in the external corpora to HASOC. The distribution of classes for Task A (NOT/HOF) and Task B (PRFN/OFFN/HATE) are reported in thousands.

However, as different corpora concentrate on various facets of hateful data, a direct one-to-one correspondence among the labels of the HASOC (listed in section 3.2) and the labels of other external datasets (*obscene* (`obs`), *identity hate* (`id.hate`), *none*, *offense* (`offn`), *hate*, *other*, *overtly/covertly agressive* (`aggr`), *profane* (`prfn`), *insult* (`ins`)) is often not possible. Thus, in most cases, we only used the external data where applicable (mostly in Task B with our BERT model). For the SVM model, we only used them in the binary (Task A) classification task. Also note that, as there was a lot of data for English, we tried to get an aggregated balanced corpora out of them. This balancing was also experimented with the Hindi training data after adding the TRAC data. Although for English the balanced set improved the results, for Hindi it was not that much helpful. We used a balanced set of around 90.2k for each of the HOF/NOT labels for our binary classification task for English.

Chapter 4

RESEARCH METHODOLOGY

This chapter will outline the technical detail and the methods used in the implementation of our system. At first, we will introduce the tasks that we are solving which include both binary and multi-label hate speech detection. After that, we will talk about the pre-processing and features that we used during the first phase of our system development. Finally, we will concentrate on our best models that make use of byte-pairs with document embeddings.

4.1 Defining the Specific Task

Given some text (e.g., tweets), we aimed to use computational methods to determine the type of hate in those texts. The task of hate speech detection can be either coarse-grained such as hate vs. non-hate (binary) or more fine-grained (multi-label). In this thesis, we tried to deal with both of the above problems of classification, which is defined as Task-A and Task-B accordingly by following the definitions from HASOC as below:

4.1.1 Task-A

Task-A focuses on Hate speech and Offensive language identification for Hindi, German and English language. This task is coarse-grained binary classification in which we tried to classify tweets into two class, namely: Hate and Offensive (HOF) and Non-Hate and offensive (NOT).

- **Non Hate-Offensive** (NOT): Posts that does not contain any hate speech or offensive content.
- **Hate and Offensive** (HOF): Post containing Hate, offensive, and profane content falls under this category.

In the additional data as well, we mapped a post as HOF if it contains any form of non-acceptable language such as hate speech, aggression, profanity etc.

4.1.2 Task-B

Task-B is the fine-grained classification. Hate-speech and offensive posts from the Task-A are further classified into the following three categories.

- **Hate speech** (HATE): Describing negative attributes or deficiencies to groups of individuals because they are members of a group (e.g. all poor people are stupid). Hateful comment toward groups because of race, political opinion, sexual orientation, gender, social status, health condition or similar.
- **Offensive** (OFFN): Posts which are degrading, dehumanizing, insulting an individual, threatening with violent acts.
- **Profanity** (PRFN): Does not involve directly insulting anyone but the usage of profane/swear words such as ass, pissed-off, fuck.

Thus we may see this problem as a 3-way (multi-label) classification task. The choice of language in this intended experiment includes English, German and Hindi. Note that we did not use additional data for this category in our SVM model.

4.2 System Architecture

The main idea of this thesis was to use a combination of the following things with the classic ML techniques such as SVM's:

- Preprocessing techniques.
- Traditional feature selection.
- Unconventional features and input representations.

We will describe each of the aforementioned components throughout this section.

4.2.1 Preprocessing

We have tried a bunch of preprocessing techniques with the initial data. Although most of the preprocessing was done for the purpose of the experiment, we only kept those that really helped improve the results in our final model. Nevertheless, here we will be discussing all the preprocessing that we have tried out.

Tokenization

Tokenization is the process of segmenting sentences or words from a document. In our experiment, we have used the word tokenizer's[1] from spaCy [58] for all the three languages.

[1] `https://spacy.io/api/tokenizer`

Lowercasing

Lowercasing refers to converting all the characters into a smaller letter. We have tried this preprocessing for Engish and German in our experiment. For Hindi, it was not applied since Hindi is a case insensitive language.

Truecasing

Truecasing is the problem of finding the proper capitalization of words within a text where such information is unavailable. For example, in English, such capitalization might occur at the beginning of a sentence or when we use proper nouns. We have only tried dealing with the first characters in a word if it's the first word in that sentence. We have used NLTK's [59] `sent_tokenize()` function[2] in order to get the sentences. Again for Hindi, it was not applicable since Hindi scripts do not have a distinction between uppercase and lowercase letters.

Normalizing URL

Since theoretically, URL's should not contribute that much on the performance, and some literature's [44] report about removing URL helps, we tried replacing them with the keyword URL. This did not help in our case, and for some languages (e.g., Hindi) it even reduced the results by fractions.

Removing Stopwords

Stopwords are the words which occur regularly in a text but do not generally contribute that much in meaning. Instead, they carry more grammatical functions. Examples of such stopwords in English are - *a*, *of*, *the* etc). At first, we tried out NLTK's default stopword list for English and German. For Hindi we sought out the stopword list[3] from spaCy. Although the default list of stopwords did not improve the results that much but by removing the *negative* polarity items (such as *no, not, nor, against* etc.) we were able to improve our results by at least 1 point in f1 score.

Stemming

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes [60]. For example the words - *compute*, *computer*, *computing*, *computation*, *computerization* all might be converted into *comput* when we apply stemming on them. Although after stemming, a stemmed token might not mean anything but it can be useful for enhancing the retrieval effectiveness, especially for text search in order to solve the mismatch problems. In our study, we have experimented with both Porter [61] and Snowball [62] stemmer from NLTK[4]. Snowball seems to be a bit better as it does not break with a big dataset.

[2] `https://www.nltk.org/api/nltk.tokenize.html`
[3] `https://github.com/explosion/spaCy/blob/master/spacy/lang/hi/stop_words.py`
[4] `https://www.nltk.org/howto/stem.html`

Lemmatization

Lemmatization usually refers to doing things properly with the use of vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma [60]. If confronted with the token saw, stemming might return just *s*, whereas lemmatization would attempt to return either *see* or *saw* depending on whether the use of the token was as a verb or a noun. In our implementation, we have tried out both the WordNet lemmatizer[5] from NLTK and spaCy lemmatizer[6], and both performed more or less similarly in terms of effectiveness.

4.2.2 Traditional Features

By traditional feature, we mean those that are commonly used with the classical machine learning techniques. These are mostly hand-crafted features that we have described in our literature (section 2.3.1 - 2.3.2). Here we will be discussing about their implementation in our system in more detail.

Count or Bag of Word

Text Analysis is a significant application field which extensively uses machine learning algorithms. However, the raw text data cannot be fed directly to these classifiers as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. The “Bag of Words" representation is the strategy that combines counting and normalization to characterize a document by word frequencies instead of taking into account the relative position and semantic orientation of the words in the text. In the case of count feature or count vectorizer, each individual tokens frequency (sometimes normalized occurrence) is treated as a feature. In our implementation, initially we have explored this feature from *scikit-learn* [63] library which is known as `CountVectorizer`[7]. The term vectorizer came from vectorization, which is is the process of converting a cluster of text documents into numerical feature vectors. Later we discarded using the count vectorizer as it performs poorly in comparison with other feature vectors (such as tf-idf).

Tf-Idf

Term frequency-inverse document frequency in short *tf-idf* is a statistical way of representing the importance of a word in relation to its document [64]. Term frequencies represent how important a term is within a document. Inverse document frequencies show how informative a term is in general, e.g., a term that occurs in very few documents is probably more informative than a term (such as *the* or *and*) that occurs in every document. To calculate the tf-idf, we take the product of the term frequency, i.e. the number of times a term occurs in a document, and the inverse document frequency,

[5] `https://www.nltk.org/_modules/nltk/stem/wordnet.html`

[6] `https://spacy.io/api/lemmatizer`

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

i.e. number of all documents given the documents that contain the term. The equation is given by:

$$tf_{term,doc} = \frac{no.\ of\ times\ term\ occurs\ in\ doc}{maximum\ occurrences}$$

$$idf_{term} = \log(\frac{no.\ of\ all\ documents}{number\ of\ documents\ that\ contains\ the\ term})$$

$$tf.idf_{term,doc} = tf_{term,doc} * idf_{term}$$

Here *maximum Occurrences* is the number of times that the most frequent term of the document occurs in the document. Here also, we have used the *scikit-learn's* implementation of `TfidfVectorizer`[8].

Word and Character n-grams

As described in section 2.3.2, we have also incorporated the n-gram model to estimate the probability of the last word/character of an n-gram given the previous words/characters. We have looked into both word and character variants of n-gram features. In the case of word n-grams, we have considered the range 1 to 3 that is *uni-grams*, *bi-grams* and *tri-grams*. As characters, we used a char n-gram with values of n ranging from 1 to 7.

4.2.3 Unconventional Features and Input Representations

By unconventional, we mean those features and input representations that people do not generally use with conventional machine learning algorithms such as SVM's.

Byte-Pair Encoding

Byte Pair Encoding (BPE) is a special kind of sub-word tokenization technique that originally falls under the category of lossless data compression algorithm [65]. It works by replacing common pairs of consecutive bytes or characters with a byte that does not appear in that data. In simple terms, it breaks down words into sub-words and forms unknown words together from smaller bits and pieces. For example, we might represent the word '*Stratford*' as '*_strat*' and '*ford*', which in terms can have an idea about the word *ford*. BPE has the following advantages over words:

- Subwords allow guessing the meaning of unknown / out-of-vocabulary (OOV) words. E.g., the suffix *-shire* in *Melfordshire* indicates a location.
- Byte-Pair Encoding gives a subword segmentation that is often good enough, without requiring tokenization or morphological analysis. In the above case the BPE segmentation might be something like - [*melf*, *ord*, *shire*].

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

- Pre-trained byte-pair embeddings work surprisingly well, while requiring no tokenization and being much smaller (in size) than alternative word embeddings.

Byte-pair embeddings are also good at learning relationships. For example, if the model learned the relationship between *old*, *older*, and *oldest*, it does not tell the model anything about the relationship between *smart*, *smarter*, and *smartest*. However, in this case, if we use some sub-tokens such as *er* and *est*, and the model learned the relationship between 'old', 'older', and 'oldest', it will tell the model some information about the relationship between 'smart', 'smarter', and 'smartest'.

In our implementation, we have used the pre-trained BPE model from BPEmb [66] library[9]. It covers almost all the major languages and is really easy to use. Although we have experimented with different *vocabulary size* as the parameter of the BPE, the best performance was achieved using a vocabulary size of *10k*.

Note that, for our subsequent experiments with the byte-pairs, we have treated them as words that is when we use word/document embeddings, we are actually using them over **sub-words** or **byte-pairs** instead of actual words.

(Sub)Word Embeddings (Word2Vec)

In order to process natural language, a mechanism for representing text is required since computers are unable to understand the concepts of words or phrases. One such representation is the word embeddings. In *word embeddings* – words from the vocabulary are mapped to vectors of real numbers. Each word is represented by a point in the embedding space and words with similar meanings are locally clustered within the space (Figure 4.1). This in terms captures some form of meaning or context about these words. Some of the most popular neural word embedding methods are the `word2vec`, `GloVe` and `FastText`.

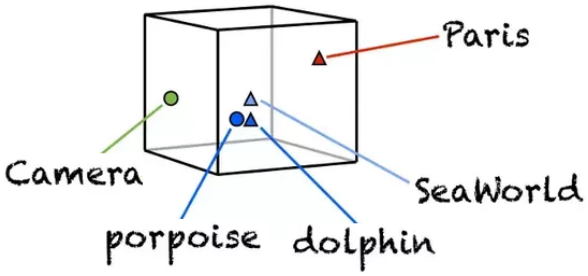


Figure 4.1: Concept of words represented by points in the embedding space.[10]

On the other hand, *bag of words* (`BoW`), *term frequency–inverse document frequency* (`Tf-Idf`) and *distributional embeddings* using co-occurrence matrix representation are three of the most commonly used traditional vector representations that are used with statistical models. While BoW and Tf-Idf do not capture contextual meaning from the surrounding words, the distributional and neural word embeddings encapsulate some form of context [67].

[9] `https://nlp.h-its.org/bpemb/`
[10] Image Source: `http://bit.ly/2B7yzxa`

For simplicity we will limit our explanation within word2vec neural embeddings which is a predictive embedding model. There are two main word2vec architectures (Figure 4.2) that are used to produce a distributed representation of words:

- **Continuous bag-of-words (CBOW)** — uses each of the contexts to predict the current word. The order of context words does not influence prediction (bag-of-words assumption).
- **Skip-gram** — model uses the current word to predict the surrounding window of context words and weighs nearby context words more heavily than more distant context words. While order is still ignored, each of the context vectors are weighed and compared independently.

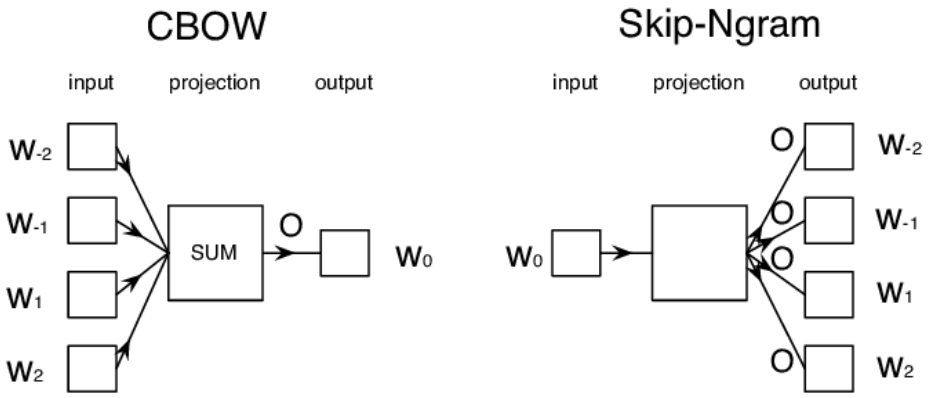


Figure 4.2: CBOW vs. skip-gram word2vec architectures.[11]

Note that, CBOW is faster while skip-gram does a better job for infrequent words.

Although, initially we started experimenting with word (actually *sub-words*) embeddings by trying to learn the embeddings using word2vec model from `HASOC` dataset (we did not use any pre-trained embeddings), at the end, we have used document embedding (*doc2vec*) for all of our models as they perform better [31] [31] [39].

Sub-word level Document Embeddings (Doc2Vec)

Doc2Vec is an extension of the Word2Vec method described above. Similarly to the idea to predict a word given its context, a paragraph will contribute to the prediction task of the word vector at the sentence level. In other words, the additional paragraph vector captures semantic properties of the sentence, which is more appropriate to the hate speech learning task at hand [31]. Formally, this paragraph vector is treated as just another word that captures these higher-level semantics.

For our implementation purpose, we are using `doc2vec`[12] from `gensim` [40] library. Like word2vec above, doc2vec also comes with two kinds of architecture - `DM` (similar to CBOW) and `DBOW` (similar to skip-gram). In our experiments, we have tested both the doc2vec flavors, and in every case, DBOW model performed better than the DM

[11] Image Source: `http://bit.ly/2B4qR72`

[12] `https://radimrehurek.com/gensim/models/doc2vec.html`

variants. We also tried combining them, but it did not give any better results than DBOW itself.

As we have stated earlier, we did not use any kind of pre-trained embeddings. Thus we trained our own document embeddings using the sub-words or byte-pairs. We used *50 epochs* to train our model with a *vector size* of *512*. Starting with a *alpha* of *0.105* we reduced the value of our alpha by *0.002* in each epoch.

4.2.4 Classifier

SVM's

For the choice of classifier, we have used `SVM` (both *linear* and *non-linear* versions). From the literature, we saw that SVM is the most popular choice among the classical ML methods and are shown to perform better for hate speech related text classification task in general. For simplicity, initially, we started our experimentation only with the binary classification task (*hate vs. non-hate*) on only English datasets. Later we considered the multi-class classification in multi-lingual settings. In all our implementation of the classifier, we have used the version of SVM[13] from scikit-learn [63] library.

A grid search over both the linear and non-linear SVM models was initially performed based on the different C and *Gamma* parameter values of the classifier. Note that we have used three different versions of the *non-linear kernels*, namely - Sigmoid, Polynomial and a Radial Basis Function (RBF) kernel in our study. In most of the cases, we have used the default values of the parameters with these kernels.

BERT

For the comparison, We used pre-trained BERT [2] models (that we have also used during the shared task) to encode a tweet into a single vector. For this, we use monolingual cased BERT-base for English (BERT_{en})[14] and German (BERT_{de})[15] as well as multilingual cased BERT (BERT_{multi})[16]. The classifier is a linear layer of depth 1, mapping the encoded tweets to labels. To deal with the unbalanced nature of the training data with this BERT model, we perform randomized weighted re-sampling of the data at each epoch. The weights given to a class is calculated such that underrepresented classes are given a larger weight and vice versa.

[13] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

[14] `https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip`

[15] `https://deepset.ai/german-bert`

[16] `https://storage.googleapis.com/bert_models/2018_11_23/multi_cased_L-12_H-768_A-12.zip`

Chapter 5

EVALUATION AND RESULTS

This chapter assesses the results produced by the feature combination and classification models described in the previous chapters. We went on experimenting with different feature selection techniques and with varying datasets combination (from Table 3.1). In all the cases, our primary evaluation was done using the HASOC corpus (Table 3.2), and we augment the training and/or the embedding data using the external corpora. During the discussion, we will divide our sections based on the Task and Languages we have evaluated. In each of these sections, we will discuss the results achieved by more traditional feature at first, followed by the results of more advanced features to show the improvement. We will also compare our results with our own BERT model (discussed in section 4.2.4) and with the models that other teams have submitted during the HASOC shared task.

5.1 Results on Task-A

For Task A, during the shared task, both BERT and SVM models, as well as an ensemble of both, were submitted. For each language, run 1 is the ensemble of all 10 folds of the top-scoring BERT model. As the recall of the NOT class is generally low, it was boosted by labelling a test sample as NOT whenever any of the folds suggested this label. For run 2, an SVM version trained on the whole dataset was submitted. Lastly, run 3 was the ensemble of all ten BERT folds and the SVM, using the same voting scheme as for run 1. Note that, we have used a 10 fold cross-validation over the HASOC training data in all of our model development and tested using the HASOC test set. Below we discuss the results on each of the languages separately.

5.1.1 English

As we have discussed about the pre-processing (section 4.2.1) and the traditional features (section 4.2.2), not all the combination of feature yields the best results. And we have experimented with numerous such combinations for each of the languages and tasks. Thus we will try to highlight mostly on our best scores during each of the sections. Nevertheless, to give an idea, in this section, we will also show the results of a few of the insignificant combinations of features that we experimented with.

Features	**Data**	**Cross (Mean)**	**Test**
tf-idf	HS_{en}	A = 0.614 F1 = 0.381	A = 0.411 F1 = 0.292
tf-idf + stem + lemma + lowercase + char n-gram(1,7)	HS_{en}	A = 0.631 F1 = 0.610	A = 0.562 F1 = 0.562
tf-idf + stem + lemma + lowercase + word n-gram(1,3)	HS_{en}	A = 0.641 F1 = 0.618	A = 0.545 F1 = 0.543
tf-idf + BPE + word n-gram(1,3)	HS_{en}	A = 0.639 F1 = 0.616	A = 0.568 F1 = 0.568
tf-idf + BPE + word n-gram(1,3) + stopword	HS_{en}	A = 0.632 F1 = 0.631	A = 0.583 F1 = 0.581
tf-idf + BPE + word n-gram(1,3) + stopword	HS_{en} +TR_{en}	A = 0.665 F1 = 0.636	A = 0.622 F1 = 0.616

Table 5.1: Scores as calculated using mean 10-fold cross validation: Accuracy (A) and F1 (micro) for Task A (NOT/HOF) for various feature combinations.

Table 5.1 shows the results of our linear SVM model (with a *C=8.5*). Here the last row shows our best combination of initial features. We also submitted the same model during HASOC shared task. In Table 5.2, we describe our results during the shared task and also compare the results with the best team as well as our own BERT model. We also show the results from our advanced features (BPE and document embeddings) in the final row.

Against Best Team	Our EN Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
YNU_{wb} **0.78/0.83**	BERT	HS_{en} +KA	–	–	0.63/0.52	0.58	0.60
	SVM_{lin}	HS_{en} +TR_{en}	–	tf-idf + BPE + word n-gram(1,3) + stopword	0.68/0.54	0.61	0.64
	Ens.	–	–	–	0.78/0.59	0.69	0.73
Doc2Vec Model	SVM_{sig}	HS_{en} +TR_{en}	$HS_{en-full}$ +TR_{en}	BPE + stopword	0.74/0.48	0.61	0.67

Table 5.2: Our submissions on the official HASOC sub task-A for English (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task A (NOT/HOF) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

From the Table 5.2, we can see that the initial score of the **SVM** model was higher

than that of the corresponding `BERT` model (*0.61* vs. *0.58*). Although our `Ensembled` model was able to get up to 0.61 in terms of f1 (macro) score, it was no-where near the best team (YNU_{wb}). On the other hand, the `doc2vec` model with sub-word level embeddings achieved a higher score than our ensembled models that we have submitted during the actual run.

5.1.2 German

We also submitted our run for the German language during the shared task. The submissions were same as with the English models. There were however, differences in the usage of external corpora. Table 5.3 illustrates the official submission scores as well as the results produced by our `doc2vec` model.

Against Best Team	Our DE Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
Hate Monitors 0.62/0.79	BERT	HS_{de} +GE	–	–	0.89/0.32	0.61	0.80
	SVM_{lin}	HS_{de} +GE	–	tf-idf + BPE + word n-gram(1,3) + stopword	0.89/0.19	0.54	0.78
	Ens.	–	–	–	0.87/0.32	0.59	0.78
Doc2Vec Model	SVM_{poly}	HS_{de} +GE	$HS_{de-full}$ +GE	BPE + stopword	**0.89/0.39**	**0.64**	**0.81**

Table 5.3: Our submissions on the official HASOC sub task-A for German (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task A (NOT/HOF) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

Based on the above results (in Table 5.3), we can see how much we have improved our results using subword level document embeddings. We are able to beat the best performing team by a margin of *0.2* on f1 macro score. Our BERT model was also really close to the winning team in terms of the scores, and stood 2^{nd} for this subtask.

5.1.3 Hindi

In terms of our Hindi model, we were able to score pretty high during the shared task. The position of our team was also 2^{nd} for this subtask as well. We can have an idea of the scores from the Table 5.4 below. From the table, we can also observe that our `doc2vec` model trained only on the HASOC Hindi corpus was able to beat our own ensembled model and was nearly as good as the top-scoring team. Note that, here we only used the HASOC training and test data for training our document embeddings.

Against Best Team	Our HI Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
Qut Nocturnal 0.81/**0.82**	BERT	$HS_{en+de+hi}$		–	0.67/0.75	0.71	0.71
	SVM_{lin}	HS_{hi} $+TR_{hi}$	–	tf-idf + BPE + word n-gram(1,3)	0.78/0.79	0.78	0.78
	Ens.	–	–	–	0.80/0.90	0.80	0.80
Doc2Vec Model	SVM_{rbf}	HS_{hi}	HS_{hi}	BPE + stopword	**0.83/0.80**	**0.81**	0.81

Table 5.4: Our submissions on the official HASOC sub task-A for Hindi (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task A (NOT/HOF) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

5.2 Results on Task-B

For task B, only BERT models were taken into consideration during the shared task. As the test data provided still contained non-hateful comments, run 1 uses the BERT ensemble from task A (run 1) to pre-select hateful comments, which are then further classified by an ensemble of the 10-folds of the top-scoring BERT model in task 2. Here, a majority vote approach was taken, such that the label with the most votes is accepted. For run 2, alternative models are trained on HS data only and were ensembled using the majority vote approach. Finally, run 3 is the ensemble of both run 1 and 2. In the case of our **doc2vec** model, we considered 3 way classification of `Profanity` (PRFN), `Offensive`(OFFN) and `Hate Speech`(HATE).

5.2.1 English

Our English model for the Task-B uses a pipeline from Task-A as described above. From the results (in Table 5.5) it is apparent that our `doc2vec` is slightly better (*0.48*) than our BERT model (*0.47*) in terms of the macro f1 score. But all of our models are way behind the winning team in this task.

5.2.2 German

In terms of German subtask B, our best model consisting of piped monolingual BERT also came 1^{st} during the competition. Using that model, we were able to get a score of *0.35* in macro f1 based measure. The results, as compared to our advanced doc2vec SVM model, is shown in the Table 5.6. In this case, however, our `doc2vec` model later was able to beat our own top scoring BERT model by a fraction of 0.01 in terms of macro f1 score.

Against Best Team	Our EN Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
3Idiots **0.54/0.75**	BERT_{pipe}	HS_{en}	–	–	0.63/0.62/0.23	0.44	0.57
	BERT	HS_{en}	–	–	0.68/0.65/0.25	0.47	0.61
	Ens.	–	–	–	0.58/0.62/0.21	0.42	0.53
Doc2Vec Model	SVM_{sig}	HS_{en}	$\text{HS}_{en-full}$ $+\text{TR}_{en}$	BPE + stopword	0.58/0.19/0.66	0.48	0.52

Table 5.5: Our submissions on the official HASOC sub task-B for English (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task B (PRFN/OFFN/HATE) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

Against Best Team	Our DE Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
LSV_UdS **0.35/0.77**	BERT_{pipe}	HS_{de}	–	–	0.89/0.17/0.18	0.35	**0.77**
	BERT	HS_{de}	–	–	0.89/0.00/0.06	0.26	0.75
	Ens.	–	–	–	0.66/0.04/0.36	0.28	0.58
Doc2Vec Model	SVM_{poly}	HS_{de}	HS_{de}	BPE + stopword	0.16/0.65/0.30	**0.36**	0.48

Table 5.6: Our submissions on the official HASOC sub task-B for German (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task B (PRFN/OFFN/HATE) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

5.2.3 Hindi

We were also able to get quite far using the unconventional features with our SVM in the case of Hindi subtask B. The winning team for this subtask had a macro f1 score of *0.58* while our doc2vec model achieved 0.57 on the same task. However, we were far behind on the weighted f1 score.

Against Best Team	Our HI Models	Train Data	Embed. Data	Features	F1	Macro	Weight.
3Idiots **0.58/0.71**	BERT_{pipe}	$\text{HS}_{en+de+hi}$		–	0.67/0.79/0.40	0.54	0.60
	BERT	$\text{HS}_{en+de+hi}$		–	0.78/0.80/0.43	0.57	0.66
	Ens.	–	–	–	0.78/0.79/0.40	0.58	0.66
Doc2Vec Model	SVM_{poly}	HS_{hi}	HS_{hi}	BPE + stopword	0.76/0.41/0.52	0.57	0.58

Table 5.7: Our submissions on the official HASOC sub task-B for Hindi (first row) against the best team (macro/weighted F1) and the Doc2vec model with Byte-Pair Embeddings (second row). The scores are shown in F1 based measure for Task B (PRFN/OFFN/HATE) as well as the average macro and weighted F1 scores. Top scoring runs are in bold.

5.3 Discussion

Throughout section 5.1 to 5.2, we have tried to compare our best models with the state-of-the-art systems. In several of the sub-task, our SVM model consisting of the **subword level document embeddings** was able to beat the top scores. From the results, we found that embedding contributes a lot on the performance of a classifier and even a statistical classifier such as SVM could benefit from the embeddings. It is beneficial in situations where there are fewer data available. To summarise, we can say that combining and optimizing the preprocessing, manual feature extraction, document embedding and byte-pair encoding; confirms the **hypothesis** that - using unconventional input representations and advanced features as part of a classical machine learning architecture leads to more accurate classification of hate speech in this dataset. We also observed that, although adding external corpora can boost the classification performance for a binary task, in the case of fine-grained detection of different types of hate, it does not benefit much from external corpora due to the incompatibility between different definitions of hate and its subtypes as well as the subjectivity of the matter.

Chapter 6

CONCLUSIONS

In this thesis, our main target was to experiment with different feature selection and input representation techniques with classical machine learning algorithms for the classification of both binary and multi-label hate speech from text. We have tried to tackle this classification task with variants of SVM classifiers. A few of the most recent related works have shown that unconventional input representation such as word-embeddings could be useful for this task even with the conventional classifiers. So far, we have not seen anyone using Byte Pair Encoding (BPE) or embeddings and sentence/document embeddings with the classical machine learning approaches in the hate speech realm. So our prime aspirations were to experiment with these features and input representations.

We were successful in combining traditional preprocessing techniques and features along with an unconventional input representation technique, namely document embeddings (doc2vec) with byte-pair encoding (BPE). Our results using the settings mentioned above showed significant overall improvements as compared to the state-of-the-art models, and we learned a lot during these experiments.

We also observed that hate speech detection is a difficult task to accomplish, and is data-driven. We also saw that not all the datasets could help achieve good results, specially because of the nature of the datasets and annotation schemes. Hate speech also heavily depends on the culture and geo-location of the text being encountered. It varies between language to language and from time to time. Presently almost all of the works and datasets rely heavily on only the English language, and there is still a shortage of gold standard datasets for even the highly resourced languages like German and Hindi. Current literature shows that classical ML approach with fine-tuned feature engineering could compete with state-of-the-art deep neural network-based models and in many ways are better. Our findings also support this hypothesis.

In a nutshell, our efforts in this thesis can be seen as:

- We have tried to implement classical ML models (using SVM's) for the task of binary (*Hate vs. Non-hate*) and multi-label (*Hate, Offensive, Profane*) hate speech classification from the text.
- We did not want to restrict ourselves only in English. Thus we planned to build multilingual models and experiment farther with feature selection and input

representation techniques for other languages such as German and Hindi.

- We have experimented with an unconventional feature (BPE) and input representation (document embeddings) that is not common with the classical machine learning approaches. We have seen improvements after using these features/representations techniques.
- We also compared our results with the results from state-of-the-art deep learning models such as BERT [68] and with other top-performing teams from the HASOC shared task.

We want to conclude our thesis with an open question for the readers -

"*Are all these classifiers that good !?*
Or is it just the way wee feed data to them ?"

Bibliography

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

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

[3] A. Tilby, *The Seven Deadly Sins: Their origin in the spiritual teaching of Evagrius the Hermit*, SPCK (2013).

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

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

[6] 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*, pages 88–93, (2016).

[7] A. Schmidt and M. Wiegand. *A survey on hate speech detection using natural language processing.* In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, (2017).

[8] D. Ruiter, M. A. Rahman, and D. Klakow, *LSV-UdS at HASOC 2019: The Problem of Defining Hate*, .

[9] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, and A. Patel. *Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages.* In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA, (2019). Association for Computing Machinery.

[10] 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).

[11] A. S. Reber, *The Penguin dictionary of psychology*, Penguin Press (1995).

[12] A. H. Buss, *The psychology of aggression*, Wiley (1961).

[13] C. A. Anderson and B. J. Bushman, *Human aggression*, Annual review of psychology **53** (2002).

[14] *Aggression - Wikipedia.* `https://en.wikipedia.org/wiki/Aggression`. (Accessed on 02/11/2020).

[15] *Sexual orientation and transgender identity hate crime - Citizens Advice.* `https://www.citizensadvice.org.uk/law-and-courts/discrimination/hate-crime/sexual-orientation-and-transgender-identity-hate-crime/`. (Accessed on 02/11/2020).

[16] S. Lipczynska, *The Greenwood Encyclopedia of Love, Courtship & Sexuality through History*, Reference Reviews (2008).

[17] *This Female Game Developer Was Harassed So Severely On Twitter She Had To Leave Her Home | Business Insider.* `https://www.businessinsider.com.au/brianna-wu-harassed-twitter-2014-10`. (Accessed on 02/12/2020).

[18] H. H. Kettrey and W. N. Laster, *Staking territory in the "World White Web" an exploration of the roles of overt and color-blind racism in maintaining racial boundaries on a popular web site*, Social Currents **1**, 257–274 (2014).

[19] G. A. Miller, *WordNet: a lexical database for English*, Communications of the ACM **38**, 39–41 (1995).

[20] 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 **10**, 215–230 (2015).

[21] J. L. Elman, *Distributed representations, simple recurrent networks, and grammatical structure*, Machine learning **7**, 195–225 (1991).

[22] 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*, pages 299–303, (2016).

[23] J. R. Firth, *The semantics of linguistic science*, Lingua **1**, 393–404 (1949).

[24] Z. S. Harris, *Distributional structure*, Word **10**, 146–162 (1954).

[25] E. Greevy and A. F. Smeaton. *Classifying racist texts using a support vector machine.* In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 468–469. ACM, (2004).

[26] 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*, pages 759–760, (2017).

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781 (2013).

[28] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, *FastText.zip: Compressing text classification models*, arXiv preprint arXiv:1612.03651 (2016).

[29] 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*, pages 745–760. Springer, (2018).

[30] J. Pennington, R. Socher, and C. D. Manning. *Glove: Global vectors for word representation.* In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, (2014).

[31] Q. Le and T. Mikolov. *Distributed representations of sentences and documents.* In *International conference on machine learning*, pages 1188–1196, (2014).

[32] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. *Recursive deep models for semantic compositionality over a sentiment treebank.* In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, (2013).

[33] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, *A convolutional neural network for modelling sentences*, arXiv preprint arXiv:1404.2188 (2014).

[34] Y. Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 (2014).

[35] H. Zhao, Z. Lu, and P. Poupart. *Self-adaptive hierarchical sentence model.* In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[36] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. *Skip-thought vectors.* In *Advances in neural information processing systems*, pages 3294–3302, (2015).

[37] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., *Universal sentence encoder*, arXiv preprint arXiv:1803.11175 (2018).

[38] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, *Supervised learning of universal sentence representations from natural language inference data*, arXiv preprint arXiv:1705.02364 (2017).

[39] 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*, pages 29–30, (2015).

[40] R. Řehůřek and P. Sojka. *Software Framework for Topic Modelling with Large Corpora*. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, (2010). ELRA. `http://is.muni.cz/publication/884893/en`.

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

[42] X. Wei, H. Lin, L. Yang, and Y. Yu, *A convolution-LSTM-based deep neural network for cross-domain MOOC forum post classification*, Information **8**, 92 (2017).

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

[44] M. Wiegand, M. Siegel, and J. Ruppenhofer, *Overview of the germeval 2018 shared task on the identification of offensive language*, (2018).

[45] M. Wiegand, A. Amann, T. Anikina, A. Azoidou, A. Borisenkov, K. Kolmorgen, I. Kröger, and C. Schäfer. *Saarland University's Participation in the GermEval Task 2018 (UdSW) – Examining Different Types of Classifiers and Features*. In J. Ruppenhofer, M. Siegel, and M. Wiegand, editors, *Proceedings of the GermEval 2018 Workshop, 14th Conference on Natural Language Processing, KONVENS 2018*, pages 21–26, Vienna, (September 21, 2018). Austrian Academy of Sciences.

[46] R. Schäfer, *Processing and querying large web corpora with the COW14 architecture*, (2015).

[47] D. Stammbach, A. Zahraei, P. Stadnikova, and D. Klakow. *Offensive Language Detection with Neural Networks for Germeval Task 2018*. In J. Ruppenhofer, M. Siegel, and M. Wiegand, editors, *Proceedings of the GermEval 2018 Workshop, 14th Conference on Natural Language Processing, KONVENS 2018*, pages 58–62, Vienna, (September 21, 2018). Austrian Academy of Sciences.

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

[49] 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*, pages 145–153, (2016).

[50] E. Raisi and B. Huang, *Cyberbullying identification using participant-vocabulary consistency*, arXiv preprint arXiv:1606.08084 (2016).

[51] I. Kwok and Y. Wang. *Locate the hate: Detecting tweets against blacks*. In *Twenty-seventh AAAI conference on artificial intelligence*, (2013).

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

[53] 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).

[54] 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*, pages 88–93, San Diego, California, (2016). Association for Computational Linguistics.

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

[56] B. van Aken, J. Risch, R. Krestel, and A. Löser. *Challenges for toxic comment classification: An in-depth error analysis*. In *Proceedings of the 2nd Workshop on Abusive Language Online, EMNLP 2018, Brussels, Belgium, October 31, 2018*, pages 33–42, (2018).

[57] R. Kumar, A. N. Reganti, A. Bhatia, and T. Maheshwari, *Aggression-annotated corpus of hindi-english code-mixed data*, arXiv preprint arXiv:1803.09402 (2018).

[58] M. Honnibal and I. Montani. *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*. To appear, (2017).

[59] E. Loper and S. Bird, *NLTK: the natural language toolkit*, arXiv preprint cs/0205028 (2002).

[60] *Stemming and lemmatization*. `https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html`. (Accessed on 02/26/2020).

[61] M. F. Porter et al., *An algorithm for suffix stripping.*, Program **14**, 130–137 (1980).

[62] M. F. Porter. *Snowball: A language for stemming algorithms*, (2001).

[63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., *Scikit-learn: Machine learning in Python*, Journal of machine learning research **12**, 2825–2830 (2011).

[64] M. Dillon. *Introduction to modern information retrieval: G. Salton and M. McGill. McGraw-Hill, New York (1983). xv+ 448 pp.,* $32.95ISBN0 - 07 - 054484 - 0$, (1983).

[65] P. Gage, *A new algorithm for data compression*, The C Users Journal **12**, 23–38 (1994).

[66] B. Heinzerling and M. Strube. *BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages*. In N. C. C. chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, (2018). European Language Resources Association (ELRA).

[67] A. Bornstein. *Beyond Word Embeddings Part 2*, (2018).

[68] 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).