HATE SPEECH DETECTION IN SOCIAL MEDIA

Kaushik Amar Das

HATE SPEECH DETECTION IN SOCIAL MEDIA

Thesis submitted to the
Indian Institute of Information Technology Guwahati
for award of the degree

of

Master of Technology

by

Kaushik Amar Das

under the supervision of

Dr. Ferdous Ahmed Barbhuiya

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY GUWAHATI

June 2020

CERTIFICATE

This is to certify that the thesis entitled **"HATE SPEECH DETECTION IN SOCIAL MEDIA"**, *submitted by* **KAUSHIK AMAR DAS** *to the Indian Institute of Information Technology Guwahati, for the award of the degree of Master of Technology, is a record of bona fide research work carried out by him under my supervision and guidance. The thesis, in my opinion, is worthy of consideration for the award of the degree of Master of Technology in accordance with the regulations of the Institute. To the best of my/our knowledge, the results embodied in the thesis have not been submitted to any other university or institute for the award of any other degree or diploma*

Dr. Ferdous Ahmed Barbhuiya,

Associate Professor and Associate Dean (R&D),

Department of Computer Science and Engineering

Indian Institute of Information Technology Guwahati

Date:

DECLARATION

I certify that

a. The work contained in this thesis is original and has been done by me under the general supervision of my supervisor(s).

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Kaushik Amar Das

ACKNOWLEDGMENTS

Firstly, I would like to thank my supervisor Dr Ferdous Ahmed Barbhuiya for taking me under his wing and allowing me to work on the thesis. He has provided me with all the necessary resources and guidance without which this thesis would not have been possible. I am also grateful to Dr Kuntal Dey who is a senior researcher at IBM for his expert advice on various topics.

I addition, I want to thank research scholar Arup Baruah for being a mentor figure and a brainstorming partner entertaining all the silly ideas that I would come up with.

And lastly, I would like to express my gratitude to my friends and family who are always there to support me.

ABSTRACT

Social Media platforms are often abused to spread hateful messages. These not only cause harm to the individual but also to society in general. The staggering volume of content generated in social media across so many countries, regions and languages make it impossible to be moderated manually. This necessitates that moderation efforts be augmented with automated tools. To this end, the thesis aims to aid this effort by developing automated hate speech detection tools. Owing to the recent success of deep learning across multiple domains, the thesis develops multiple deep learning models for detecting hate speech in social media. The thesis develops various such models using DNN architectures like CNN, BiLSTM to the more recent BERT-based state-of-the-art pre-trained models. These models are evaluated using datasets not only in English but also in low resource languages such as Indian Bengali, Hindi and their code-mixed variants. These datasets are collected from various sources like Facebook, Twitter and YouTube. In addition, the thesis also studies the detection of online aggression and hate speech identification in internet memes.

List of Abbreviations

- **A**
  - **ANN**: Artificial Neural Networks
  - **AUC**: Area Under the Curve
  - **AI**: Artificial Intelligence
- **B**
  - **BERT**: Bidirectional Encoder Representations from Transformers
  - **Bi-LSTM**: Bi-directional Long Short Term Memory
- **C**
  - **CBOW**: Continuous Bag Of Words
  - **CNN**: Convolutional Neural Network
  - **COLING**: Conference of Computational Linguistics
- **D**
  - **DNN**: Deep Neural Network
- **F**
  - **FIRE**:Forum for Information Retrieval Evaluation
- **H**
  - **HASOC**: Hate Speech and Offensive Content Identification in Indo European Languages
- **L**
  - **LREC**: Language Resources and Evaluation Conference
  - **LSTM**: Long Short Term Memory
- **N**
  - **NLP**: Natural Language Processing
- **O**
  - **OCR**: Optical character recognition
- **P**
  - **POS**: Parts of Speech
- **S**
  - **SemEval**: International Workshop on Semantic Evaluation
- **T**
  - **TF-IDF**: Term Frequency - Inverse Document Frequency
  - **TRAC**: Trolling, Aggression and Cyberbullying
- **U**
  - **USE**: Universal Sentence Encoder

Contents

Table of Contents **xv**

List of Figures **xvii**

List of Tables **xix**

1 Introduction **1**
1.1 Objective . . . 2
1.2 Challenges . . . 3
1.3 Contributions . . . 5

2 Literature Review **7**
2.1 Datasets . . . 7
2.2 Related Works . . . 9

3 Detecting Hate Speech in Indo-European Languages **15**
3.1 Introduction . . . 15
3.2 HASOC-2019 Dataset and Classification Tasks . . . 16
3.3 Methodology . . . 20
3.3.1 Embeddings . . . 20
3.3.2 Models Used . . . 21
3.4 Results . . . 23
3.5 Error Analysis . . . 24
3.6 Discussion and Conclusion . . . 26

4 Checkpoint Ensemble of Transformers for Hate Speech Detection **27**
4.1 Introduction . . . 27
4.2 OffensEval-2020 Dataset and Classification Sub-Tasks . . . 28
4.3 Checkpoint Ensembling . . . 29
4.4 Transformer Models . . . 32
4.5 Classifiers . . . 32
4.5.1 Sub-Task A: Ensembled GPT-2 . . . 33
4.5.2 Sub-Task B: Ensembled RoBERTa . . . 33
4.5.3 Sub-Task C: Ensembled DistilRoBERTa with Attention Mask Dropout . . . 34
4.6 Results and Error Analysis . . . 34
4.7 Discussion and Conclusion . . . 36

5 Automated Aggression Identification using Transformers **37**
5.1 TRAC-2 Dataset and Classification Sub-Tasks . . . 37
5.2 Methodology . . . 40
5.2.1 Attention Head Layer . . . 41

5.2.2 Classifiers . . . 42
5.3 Results and Error Analysis . . . 43
5.3.1 English . . . 43
5.3.2 Hindi . . . 45
5.3.3 Bengali . . . 46
5.4 Discussion and Conclusion . . . 47

6 Using Text and Image Features to Classify Internet Memes 49
6.1 Introduction . . . 49
6.2 Memotion-2020 Dataset and Classification Tasks . . . 50
6.3 Methodology . . . 53
6.3.1 Pre-trained Models . . . 53
6.3.2 Classification Systems . . . 53
6.3.3 System Training and Inference . . . 55
6.4 Result and Error Analysis . . . 56
6.5 Discussion And Conclusion . . . 58

7 Conclusion 61
7.1 Future Work . . . 62

Bibliography 63

Author's Biography 73

List of Figures

1.1 Hateful Meme Examples 4

3.1 Train Set Countplots 18
3.2 Train Set Countplots 19
3.3 Stacked BiLSTM Architecture 21
3.4 CNN Architecture 22
3.5 Confusion Matrices For Sub-Task A 24
3.6 Confusion Matrices For Sub-Task B 25
3.7 Confusion Matrices For Sub-Task C 26

4.1 OffensEval-2020 English Dataset 29
4.2 OffensEval-2020 English Test Dataset 29
4.3 Diagram based on [1]. The square boxes are models at each step of a training process. Lighter shade means better performance. In traditional ensemble (4.3b), we pick the best model found during the training process. In checkpoint ensemble (4.3a), we pick the different checkpoints of the same model. 30
4.4 Transfer Learning Model 32
4.5 Confusion Matrices 36

5.1 Train Set Countplots 39
5.2 Test Set Countplots 40
5.3 Classifier Architecture 42
5.4 English Test Set Confusion Matrices 44
5.5 Hindi Test Set Confusion Matrices 45
5.6 Bengali Test Set Confusion Matrices 46

6.1 Search popularity of keyword ‘meme’ via Google Trends. A value of 100 indicates peak popularity 50
6.2 Memotion Dataset Countplots 52
6.3 Classifier System Overview. System 1 uses both of the pre-trained model. System 2 only makes use of the images (colored in blue). 54
6.4 Meme Examples 57
6.5 Misclassification Errors 57
6.6 Test Set Confusion Matrices 58

List of Tables

3.1 Embeddings used for each language . . . 21
3.2 Stacked BiLSTM Architecture . . . 22
3.3 CNN Architecture . . . 23
3.4 Official Results . . . 24

4.1 Official Test Set Results . . . 34
4.2 Detailed Test Set Performance . . . 35

5.1 TRAC-2 Dataset Statistics . . . 38
5.2 Official Test Set Results. Hindi Sub-Task B obtained rank 2 . . . 43
5.3 English Dev and Test Set Results . . . 43
5.4 Hindi Dev and Test Set Results . . . 45
5.5 Indian Bengali Dev and Test Set Results . . . 46

6.1 Task A Label Counts . . . 51
6.2 Task B Label Counts. The Samples marked 0 are negative for their corresponding label . 51
6.3 Different image formats in the dataset . . . 51
6.4 Memotion Dataset Statistics . . . 51
6.5 Model Hyper-parameters . . . 55
6.6 Experiment Results . . . 56
6.7 Official Test Set Results . . . 56

Chapter 1

Introduction

Social media is arguably one of the most used interpersonal communication systems today. It connects people to their friends, relatives, communities of interest and also lets them discover new ones or even create one if they so desire. This computer-mediated [2] medium of interaction has given a platform to just about anyone with an internet connection to voice their opinion about anything, anytime. Micro-blogging platforms such as Facebook and Twitter are notoriously popular in this regard.

Unfortunately, these platforms are often abused for malicious purposes such as disparaging communities, spreading hate and cyber-bullying. These have caused irreparable damage to society and to the individual. For example, hate speech spread on Facebook incited the genocide of Rohingya people in Myanmar [3]. Sri Lanka had to shut down Facebook, WhatsApp and Instagram to stop anti-Muslim violence [4].

To fight off the online spread of harmful content, countries have started introducing new laws. Germany enacted a bill in 2018 [5] that mandates social networking sites to swiftly remove hateful content from their site within 24 hours. Similar legislation was also passed in France's lower house in July 2019 [6]. Social media platform are also constantly updating and improving their content policies to make their platforms safer. But legislation and policy alone cannot curb this problem. In addition, considering the vast amount of social media posts made each day, manual moderation is infeasible. It requires automated techniques that can quickly and correctly identify such hazardous content and that too at scale.

Detecting hate speech across the millions of social media posts made every day across hundreds of languages, regions and countries is no trivial task. Currently, AI combined with human reviewers is at the front-line of the battle against hate speech [7]. Although much progress has been made, there will never be a perfect system. Hence, in light of this monumental task, the aim of this thesis is to aid this fight against hate speech.

1.1 Objective

The primary objective of this thesis is the design and development of automated techniques for detecting hate speech in social media. It can be summarized simply as follows:

† Given a post in social media, classify is as either *Hate speech* or *Harmless*.

This begs the question as to what *is* hate speech and how it is *defined*. One of the earliest definitions of hate speech is given by Nockleby et al. [8] in 2009. This definition states that hate speech is *any communication that disparages a person or a group on the basis of some characteristic such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic*.

Twitter's policy [1] considers hate speech as the *promotion of violence against or directly attack or threaten other people based on race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease*.

Davidson et al (2017) [9] state that *language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group* is hate speech.

Hate Speech Examples

- I don't know how much more I can take! 45 is a compulsive liar! #Trump30Hours #TrumpIsATraitor
- Fck the US tho and #f**ktrump https://t.co/Wyc6RQW02x
- @KurtSchlichter LEGAL is. Not illegal. #BuildThatWall
- This immigrant should be hung or shot! Period! Animal.

However, the above-mentioned objective is not inclusive enough in a broader sense of the current state of social media. Firstly because hate speech can take many forms. Hateful content is also spread as videos and pictures and is not restricted to simple text. Secondly, selectively fighting only against hate speech disregards other types of content in social media that are equally appalling and detrimental to society, such as aggression and misogyny. As per the definitions above, these categories intersect with our primary objective. Therefore, while the primary objective of the thesis remains the same, it also studies the detection of online aggression. And the identification of hate speech in internet memes.

[1] https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

1.2 Challenges

Detection of hate speech is a research problem that has its own set of unique challenges. It can be argued that detecting hate speech is not as simple as trivial keyword spotting. Some of the properties [10] of hate speech that makes its detection a challenging task are listed below.

1. **Hate changes with time**. What constitutes hate speech changes with time. For example, hate speakers use the hashtag ”#buildthewall” to target immigrants. However, this hashtag was not used in the context of immigrants before Donald Trump’s presidency. Also, there are too many permutations and combinations that one can use to obfuscate abusive words. For example, one could obfuscate ”n*gger” into ”n1993r”.

2. **Context is hard to get**. It is not always clear to understand the context of the words from short documents such as tweets. African Americans use the word ”n**ga” in everyday casual language. This racial term only becomes hate speech when it is used by people of other races. It is not possible to determine whether a person of colour or whether a person of some other ethnicity is using the slur by simply looking at a tweet.

3. **There are too many slurs and insults to keep track of.** It is reasonable to expect that there must be a comprehensive list of all the profane terms used for the expression of hate speech. However, the unfortunate truth is that new terms keep popping up with time. It is difficult to keep track of the changes. Also, what might be offensive or unacceptable to some group might be acceptable to some other group.

4. **It is not always noisy.** There are many instances where the hate speech posted is noisy and incoherent. For example, ”*F**king jews maggots with tiny a$$hats*” But there are also exceptions where hate speech is expressed using fluent and grammatically correct semantics. As such it is difficult to generalize on a pattern that can be used to express hate speech, such as, ”*Immigrants need to be shot on sight. Stop wasting tax money on border patrol*”.

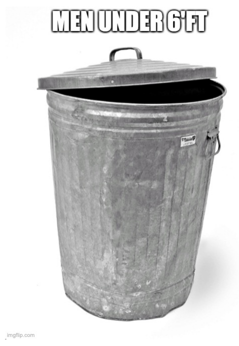
5. **It can cross sentence boundaries.** The actual hate speech might not be directly obvious from the initial sentences of a post. They could be anywhere in the post and it is not always obvious where. For example, ”*All women should be loved and cherished. However, everyone should avoid coon poons.*” The hate in this example is in the second sentence which, but required an understanding of the terms used to identify it as hate speech. Also, the first sentence is excessively positive and might trick machine learning classifiers into labelling it as harmless.

6. **Differentiating between sarcasm and abusiveness.** Posts like, ”*Yes my friends, let’s make America great again by sending back all the tacos!*” are sarcasm. But one might confuse it as hate speech against immigrants. Classifying this required knowledge about real-world events and communities.

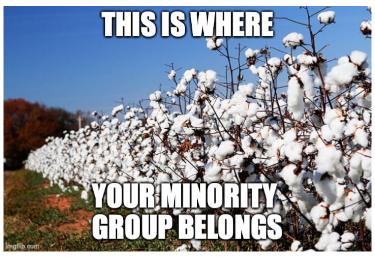
7. **It might be multi-lingual and code-mixed.** In a diverse country such as India, people often use English alongside their vernacular language, which is referred to as code-mixing. For example,“ *shut the f**k up idiot.* भोसडीके *kuch bhi bolta rehta hai* ”. As such, understanding of multiple languages might be necessary.

8. **Dearth of Annotated Data**. The ever-changing nature of the problem means that data gets outdated and might no longer be useful for training classifiers. Also, some examples of hate speech are so complicated that even expert annotators are unable to label them.

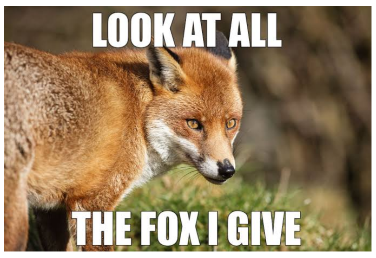
9. **It is not restricted to text only.** Much of the hate speech today is also spread in videos and images. This doubles the challenge as it might require understanding the context from multiple domains. For example, the meme in Fig 1.1a uses the image of the trash can to imply that men under 6ft are garbage. The meme in Fig 1.1b is offensive towards the African American community. But, one will not be able to identify the hateful message in this example without historical knowledge of the time when African American people were used as slaves in cotton plantations [2]. And the meme in Fig 1.1c uses the image of a fox to imply the cuss word “f**k”.



(a) Sexist



(b) Racist



(c) Offensive

Figure 1.1: Hateful Meme Examples

At the time of writing this thesis, the current state of the literature suggests that the best way to meet all the challenges above is through deep learning. This is because, the thesis primarily deals with text classification problems, which is a part of natural language processing. And the

[2] https://en.wikipedia.org/wiki/Slavery_in_the_United_States

current state-of-the-art in natural language processing is held by deep learning. Starting with the Transformer architecture [11] in 2017 to BERT in 2019 [12], deep learning holds the highest scores in multiple NLP benchmarks, ranging from question answering to sentiment analysis. It has also found success in a lot of other domains such as speech and computer vision [13]. Additionally, deep learning has the benefit of requiring little to no feature engineering. Social media texts can be inherently noisy due to spelling errors, code-mixing, acronyms and poor grammar. This makes the extraction of linguistic features using NLP tools especially difficult. These reasons motivated the thesis to pursue deep learning as a tool for the task of detecting hate speech.

1.3 Contributions

The thesis starts off by reviewing the pre-existing literature in the field of hate speech detection in Chapter 2. In addition to summarizing relevant research papers in the field, the review also discusses the prominent datasets used for evaluating hate speech detection techniques.

Chapter 3 proposes two deep learning based classifiers for detecting hate speech in the code-mixed Hindi, English and German Languages. The thesis experiments with oversampling techniques to mitigate class imbalance.

The thesis then moves on to experimenting with ensembling and pre-training techniques like BERT in Chapter 4. To save time and quickly create ensembles, the thesis proposes the usage of *Checkpoint Ensembling*. This chapter also deals with the challenge of training on a massive English dataset.

In parallel with Chapter 4, the thesis also builds models for detecting online aggression, trolling and cyber-bullying in Chapter 5. The proposed RoBERTA based ensemble model in this chapter ranked 2^{nd} amongst all the models evaluated on the TRAC-2 dataset.

Lastly, before concluding in Chapter 7, the thesis works on the detection of offensive, sarcasm, motivation and humour in internet memes in Chapter 6. For this, the thesis develops two different systems, one which uses text and images feature of the meme, while the other uses only the image. This chapter also analyses the strengths and weaknesses of both of these systems.

To the best of our knowledge, the contributions of the thesis are novel. This is because all the datasets used in this thesis were proposed recently in the sense that there did not exist any previous works on them at the time when their corresponding chapters were being written in this thesis. They were released as part of different research workshops during 2019-2020. Many research works, including this thesis, were being carried out simultaneously on these datasets.

As such, for each of the classification models proposed in this thesis, a rank of the model is also mentioned. These ranked were obtained after evaluating unlabelled test data using the models proposed in this thesis. This evaluation was done by submitting a model's predictions on an unlabelled test set to the dataset authors themselves. In the absence of any previous works, this rank serves as a competitive comparison of the approaches proposed by the thesis versus research works throughout the world. Some of these other works are discussed in Chapter 2.

At the end of each chapter, a Github repository containing the code for the experiments and proposed models is also given for further reference. Although much work has been done to make the code as simple as possible, it requires some knowledge of Python and deep learning frameworks such as PyTorch and Keras to understand.

Chapter 2

Literature Review

The field of Hate Speech Detection has been constantly evolving, with new research being published consistently for the last few years. In this chapter, the thesis covers some of the related works that have been done previously. §2.1 describes the most prominently cited datasets in this field. §2.2 discusses the related works.

2.1 Datasets

1. **Djuric et al.** (2015) [14] collected a large dataset from the Yahoo! Finance Website. This dataset contained $951,736$ comments by $209,776$ anonymised users. These comments were collected and labeled over a period of six months. This dataset contained $56,280$ hate speech, and $895,456$ clean comments. This dataset is not public.

2. **Waseem et al.** (2016) [15] retrieved $136,052$ tweets from Twitter, out of which they annotated $16,914$ tweets. These collection of tweets contained $3,383$ sexist, $1,972$ racist and $11,559$ neither sexist or racist tweets. These tweets were sent by 614 different users. This dataset is publicly available [1].

3. **Nobata et al.** (2016) [10], over the period between October 2012 and January 2014 collected a large dataset of user comments posted on Yahoo! Finance and News. Annotators were asked to categorize 10% of all the comments posted each day into two types, clean and abusive. The finance portion of the dataset contained $448,436$ comments out of which $15,181$ are abusive and $433,255$ are clean. The news portion contained a total of $726,073$ comments, out of which $70,311$ were abusive and $655,732$ were clean.

[1] https://github.com/zeerakw/hatespeech

4. **Davidson et al.** (2017) [9] used the Twitter API to sample $33,458$ tweets and had them manually labelled by CrowdFlower (CF) workers. The workers labelled the tweets into three types: a) *hate speech*, b) *offensive but not hate speech* and c) *neither offensive nor hate speech*. Each tweet was labelled by three or more workers and the majority decision for each tweet was assigned as the final label. When a majority could not be reached, the tweet was discarded, finally leaving a sample of $24,802$ tweets, which they have made publicly available [2].

5. **Kumar et al.** (2018) [16] compiled a dataset for the task of Aggression Detection in social media as part of the TRAC workshop competition. It was annotated with three levels of aggression: Overt, Covert and Non-Aggressive. This dataset contained $12,000$ randomly sampled comments from Facebook as the training set and an additional $3,000$ comments as the validation set. Each of which was made available in two different languages, English and Hindi. The English test set contained 916 tweets while the Hindi test set comprised of 970. They also gave additional $1,257$ English and $1,194$ Hindi Tweets as a *surprise test set* to the participants. This dataset is publicly available online[3].

6. **Basile et al.** (2019) [17] organised a workshop (*HatEval*) for the task detecting hate speech against women and immigrants in Twitter. Their entire dataset contained a total of $19,600$ tweets, of which $13,000$ were English and $6,600$ are Spanish. Out of them, $10,509$ are targeted towards women and $9,091$ are targetted towards immigrants. They collected their data from Twitter using the following strategies.

   (a) Monitor potential victims of hate accounts.

   (b) Fetch history of identified haters

   (c) Filter Twitter streams for keywords, i.e words, hashtags, and stems.

7. **Sabat et al.** (2019) collected a dataset of hateful internet memes. This dataset contained $5,020$ memes which were weakly labelled as either hate or non-hate. These hateful memes were downloaded from Google Images using the following queries: racist meme (643), Jew meme (551), and Muslim meme (501). In this way, they collected a total of 3,325 hateful memes. The non-hate memes were sourced from a Reddit Memes Dataset [4]. This dataset is available online[5].

[2] https://github.com/t-davidson/hate-speech-and-offensive-language
[3] https://sites.google.com/view/trac1/shared-task
[4] https://www.kaggle.com/sayangoswami/reddit-memes-dataset
[5] https://github.com/imatge-upc/hate-speech-detection

2.2 Related Works

- Djuric et al. (2015) [14] developed a two-step method for hate speech detection. Firstly they created a low-dimensional text embedding using paragraph2vec [18]. Secondly, they train a binary classifier to distinguish between hateful and clean comments. Paragraph2vec is an unsupervised algorithm that learns to represent variable-length pieces of text, such as sentences and paragraphs as fixed-length feature representations. They use this algorithm to jointly model comments and words in a distributed representation using the continuous Bag of Words (CBOW) neural language model. A CBOW neural language model tries to predict the target word according to its context. This context is represented as a bag of words contained in a fixed window around the target word. For example, given the sentence '*Like a bird it flew away into the sky* and' the target word '*bird*', a CBOW model would try to predict '*bird*' using the surround context window '$\{$*like,a,it,flew*$\}$'. The model would sum the vectors of the context to predict the target. Once they had trained and validated their paragraph2vec representations, they trained a logistic regression classifier. They outperformed previous methods and achieved the AUC (Area Under the Curve) score of 0.8007. Their dataset is described in §2.1.

- Nobata et al. (2016) [10] studied the performance of various NLP features in detecting hate speech using a regression-based classifier [6]. The NLP features were chosen such that they represent different aspects of a document. They experimented with a range of different features which can be broadly categorized into the following four types:

  1. *N-grams*: They used character n-grams of 3 to 5 characters including spaces and token unigrams and bigrams.
  2. Linguistic features: These included things like the length of comment in tokens, the average length of words, presence of inflammatory words, presence of polite words, etc.
  3. Syntactic Features: They used ClearNLP v2.0 dependency parser [7] to extract features such as POS (Parts of Speech) tags, dependency relations,etc. These features are supposed to capture long-range dependencies between words that n-grams might not be able to.
  4. Distributed Features: These included vector space representations [19]. They tested

[6]https://github.com/JohnLangford/vowpal_wabbit
[7]https://emorynlp.github.io/nlp4j/

three such representations namely word2vec[8], pretrained embedding[9], and comment2vec [18].

- Davidson et al. (2017) [9] work focused on evaluating the performance of various machine learning techniques for the task of detecting hate speech in tweets. They had extracted a variety of features to train their machine learning models. After lower-casing and stemming their data with Porter stemmer, they create 1-gram,2-gram and 3-gram features, each of which is weighted by its TF-IDF. Syntactic features are also extracted using NLTK [20]. The quality of tweets are quantified. Sentiment scores are assigned to each tweet using a sentiment lexicon. Finally, they include binary and count indicators for hashtags, mentions, retweets, etc. They applied various machine algorithms which include logistic regression, naïve bayes, decision trees, random forests, and linear SVMs. Their best performing model is logistic regression with L2 regularization.

- Malmasi et al. (2017) [21] used the dataset of Davidson et al. and investigated the performance of classification systems in discriminating between profanity that is used for expressing hate speech and casual context. The features that they have used in their study included surface n-grams, word skip-grams and word representation n-grams [22] and brown clusters [23]. In their study, they train 16 linear SVMs to evaluate how each of their features performed. They also experimented with a single classifier trained using a combination of all of the aforementioned features (5.5 million in total). However, they found that this single classifier did not improve the performance of their tasks. They also studied various ensemble methods. One of their ensembles combined multiple classifiers using meta-learning. Meta-learning learns from the knowledge learned by other classification algorithms to improve performance. Their RBF-Kernel SVM meta-classifier improved upon the previous best and achieved an accuracy score of 79.8%.

- Gamback et al. (2017) [24] utilized convolutional neural network (CNN) based architectures to classify the hate speech dataset by Waseem et al. [15] They trained four different CNNs on four different features, namely character 4-grams, word vectors using word2vec [8], randomly generated word vectors, and word vectors combined with character n-grams. They found that word2vec embeddings performed best achieving an F1-score of 78.3%.

- Gröndahl et al. (2018) [25] replicated seven state-of-the-art models for hate speech detection and studied how these models can be fooled into labelling hate speech as harmless.

[8]https://code.google.com/p/word2vec/
[9]https://github.com/turian/crfchunking-with-wordrepresentations

They used six types of evasion attacks that can easily fool classifiers that rely on surface-level features, including Google's toxicity indicator Perspective API. These attacks are described as follows:

- Word Changes
  1. Inserting Typos. This attack involved modifying the words such that it would reduce the likelihood detection as hate speech, avoid being corrected by spell checkers and also retain its original meaning.
  2. Leetspeak. This simply meant replacing alphabets with visually similar characters, such as replacing *i* with *1* , *s* with *5*.
- Word appending
  1. Appending common words. Here the attacker simply appends commonly appearing English words.
  2. Appending non-hate words. Same as above, but these words are used more commonly in non-hate contexts
- Word boundary changes
  1. Inserting white spaces. Here the attacker inserts additional separators between words to reduce the classifiers' ability to tokenize words correctly.
  2. Removing white spaces. Here the removes white space to achieve the same effect as above.

They proposed an adversarial training scheme to mitigate against said attacks. They found that models trained using character-level features are more resistant.

- Aroyehun and Gelbukh (2018) [26] in their work in detecting aggression in social media, showed that deep learning methods benefit from more data compared to traditional machine learning methods. They increase the amount of data by they had by translating the training samples into 5 different languages and then translating them back into their original language, resulting in a 5 fold increase. They also augmented their training data with a related hate speech dataset. They used the best performing model on their training set to label the samples in the hate speech dataset. These loosely labelled samples were then added to their aggression dataset. They found that such augmentation techniques help to generalize the models and correctly classify data from unseen sources. They studied the performance of their augmented dataset using various CNN and LSTM [27] based architectures.

- Arroy-Fernández et al. (2018) [28] explored the usage of Naïve Bayes, Linear Perceptron, Passive-Aggressive classifier, SVM and a fusion of two different classifiers to detect cyberbullying in social media. Their fusion classifier was simple, if two classifiers agree on the predicted label, it is assigned to the sample. Otherwise, assign a statistically probable label using by a third classifier. When converting text into vector space representations, they found that TF-IDF to be more stable and accurate compared to hashing (bag of words) and word-embedding representations. They argue that word embedding representations are too complex which leads to over-fitting of the classifiers.

- Raiyani et al. (2018) [29] found that a simple dense architecture can outperform various other deep learning models in the task of detecting aggression in social media. Their dense model architecture consists of just three dense layers of size $1024, 512, 256$ respectively. Their model simply made use of one-hot-encoded representations of the input samples. This simple approach performed better than more complex methods such as LSTM, CNN, FastText and voting classifiers. They noted that the activation functions used for each layer made a large impact on model performance. In addition they also attempted to denoise their dataset using various pre-defined rules which included rules such as undoing abbreviations (*luv* $\rightarrow$ *love*) and reframing emojis (:-) $\rightarrow$ em_smile), etc.

- Rizoiu et al. (2019) [30] research focused on three key areas for the task of detecting hate speech. Firstly, designing a general-purpose hate embedding, which could be used out the box thereby negating the need for expensive handcrafted features. Secondly, leveraging multiple unrelated datasets to jointly learn and transfer knowledge between unrelated learning tasks. And thirdly constructing a tool for separating the different types of hate speech, and characterizing the different types of hate speech. For the first area, they develop a deep neural network (DNN) architecture called *DeepHate* that whose primary utility was to create task-specific word and sentence embeddings. They work the second area, by developing a DNN architecture called *t-DeepHate* using transfer learning methods that leverage smaller unrelated data sets to train an embedding capable of representing ”general purpose” hate speech. And for the third area, they create *Map Of Hate*, which is a 2D visualization tool for hateful context, capable of differentiating between different types of hateful content and providing an explanation for what makes text hateful.

- Baruah et al. (2019) [31] used Bi-directional Long Short-term Memory (BiLSTM) [27] based architecture to identify hateful tweets in the HatEval datset [17]. Their work compared Bi-LSTM models with and without attention [32] mechanism and logistic regression (LR). They used 200-dimensional pre-trained Glove Embeddings [33] for converting text into vector space representations. These vectors are used as input to their BiLSTM networks. They found that BiLSTM networks without attention performed better than the one used with an attention mechanism. Their model achieved an f1-score of 0.51 on the test set, obtaining an official ranking of 8 (out of 71) on the dataset.

- Indurthi et al. (2019) [34] also worked on the same task as Baruah et al. (2019) mentioned above. They evaluate the performance of various sentence embeddings and how effectively they can be used with off-the-shelf machine learning algorithms. They evaluate the performance of various sentence level embeddings such as InferSent [35], Concatenated Power Mean Word Embedding [36], Lexical Vectors [37], Universal Sentence Encoder [38], Deep Contextualized Word Representations(ELMO) [39]. They balanced the dataset using Synthetic Minority Over-sampling Technique (SMOTE) [40]. Their model utilizing Universal Sentence Encoder with support vector classifier with RBF kernel yielded 0.65 f1-score, which is the highest score in the HatEval dataset.

- Mishra and Mishra (2019) [41] utilized pre-trained BERT transformers for the fine-grained and binary classification of hate speech. They fine-tune pre-trained BERT models using a joint-label training method. In this approach, they create new classes by forming all possible class combinations across multiple classification tasks via their class labels. This approach enables the model to shares information between different tasks and makes it possible to train only a single model for multiple tasks. Additionally, join-label training addresses data imbalance issues across class labels as the full training data is used to solve multiple tasks simultaneously. Their approach is one of the top performers on the HASOC dataset [42]. This model was also successfully used for aggression detection [43].

- Sabat et al. (2019) [44] built three different models for classification of hate in internet memes. Of the three models, one used only text, one used only image and the last one used both text and image together. They used Google Tesseract 4.0 OCR for extracting the text from the memes. BERT was used to extract features from the text and VGG-16 was used to extract visual features from the images. These features were concatenated together and use as input to train a multi-layer perception classifier. They found that this multi-modal model outperformed their other two uni-modal models which used only text or image.

- Risch and Krestel (2020) [45] proposed an ensemble of pre-trained BERT transformers using bagging to overcome high variance problems during fine-tuning. BERT being a large transformer can be easily fine-tuned across many tasks, but it can also overfit just as easily. This over-fitting effect is more prominently seen in smaller datasets. They overcome this issue by training many models, each with a different chunk of the training data and a different random initialization of the classification layer. Their study reveals that the performance of multiple BERT models is always greater than a single model. They show that performance increases up to 15 models after which it starts to plateau. They noted that different random initialization plays a bigger role than varying training sets while creating such ensembles. Their approach achieved the best weighted-f1 score in five out of six shared tasks of the TRAC-2 dataset [46].

- Kumar et al. (2020) [47] proposed a hybrid ConvNet-SVM_{BoVW} model for performing sentiment analysis on multi-modal social media data such as an internet meme. This proposed model is a combination of four modules which are sequenced as the discretization, text analytics, image analytics, and decision module. The first module separates text from the image using Google Lens. The second module applies a context enriched convolutional neural network to determine the sentiment of the text. Then the third module uses an SVM trained on bag-of-visual-words predicts the sentiment of the image. And finally, the decision module uses Boolean operations to make the final prediction.

- D. M. Beskow et al. (2020) [48] evaluated various uni-modal and multi-modal approaches for classifying images on the internet as either memes or non-memes. To this end, they collected approximately 25,000 memes and the same amount of non-memes. The memes were collected from various sites such as Flickr, Reddit, Twitter and Facebook. The non-memes were randomly scraped from Google Images and Twitter. For training their systems, they used different types of features such as text, text length, image, facial features. Using these features they studied a variety of different models which included logistic regression, Naïve Bayes, CNN, LSTM, Inception-V3, VGG18, etc. Their study found that the multi-modal approaches only provided small gains in performance compared to a image-only uni-modal approach.

Chapter 3

Detecting Hate Speech in Indo-European Languages

3.1 Introduction

Most of the research work on hate speech detection has been particularly focused on the English language. The challenges posed by Indo-European languages have not been addressed as widely compared to that of the major languages. To stimulate research, develop data and evaluation resources on other languages the HASOC 2019 [49] Workshop released a new dataset for the Identification of Offensive content in Indo-European languages.

In this chapter, the thesis develops a Stacked BiLSTM and a CNN based model for the aforementioned task. The models use pre-trained Glove and FastText embeddings for converting text into a vector representation. These models are first trained to do a binary classification of hate speech vs harmless. And then are fine-tuned for fine grained classification of sub types of hate speech such as targeted and untargeted.

This chapter is organized as follows. In §3.2, the thesis describes the HASOC-2019 dataset and the classification tasks it features. The classifier models and their methodology is described in §3.3. Their performance is discussed in §3.4. The errors made by the models are explored in §3.5 and finally, the chapter is concluded in §3.6.

3.2 HASOC-2019 Dataset and Classification Tasks

The HASOC dataset is annotated for three classification tasks which it's creators refers to as sub-tasks A, B and C. Except for C, each of the three sub-tasks are featured in the Hindi, English and German languages. Sub-task A comprised of classifying the document into two categories, *Hateful and Offensive* and *Non - Hateful and Offensive*. The items classified as hateful and offensive in sub-task A are to be further classified into three fine-grained classes namely *Hate speech*, *Offensive*, *Profane* for sub-task B. Sub-task C, further classifies the hateful and offensive items of sub-task A into two types of offences, *Targeted Insult* and *Untargeted*. The datasets are compiled from Facebook and Twitter posts. The definitions for each label are given *verbatim* in the List 3.2. The following definitions were followed wherever applicable by the annotators:

- **Hate Speech**: 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**: Posts which are degrading, dehumanizing, insulting an individual, threatening with violent acts.
- **Profanity**: Unacceptable language in the absence of insults and abuse. This typically concerns the usage of swearwords (Scheiße, Fuck etc.) and cursing (Zur Hölle! Verdammt! etc.) are categorized into this category.

HASOC 2019 Sub-tasks

A) *Hate speech and Offensive language identification*

- Non Hate-Offensive (NOT) : This post does not contain any Hate speech, offensive content.
- Hate and Offensive (HOF) : This post contains Hate, offensive, and profane content.

A post is labelled as HOF if it contains any form of non-acceptable language such as hate speech, aggression, profanity otherwise NOT.

B) *Fine-grained Classification of Hate-speech and Offensive posts*

- Hate Speech (HATE) : Posts under this class contain Hate speech content.
- Offensive (OFFN) : Posts under this class contain offensive content.
- Profane (PRFN): These posts contain profane words.

Posts containing general profanity are not targeted, but they contain non-acceptable language.

C) *Offense Target Identification*

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

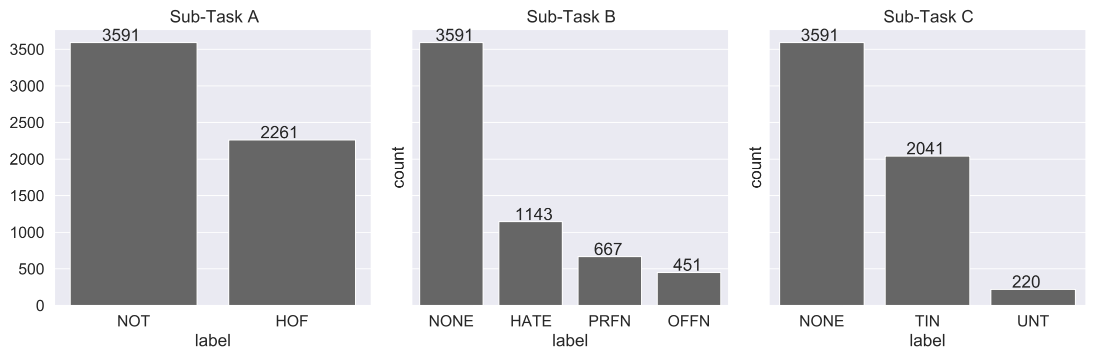
Only posts labeled as HOF in sub-task A are included in sub-task C.

The number of training samples provided per class for English, German and code-mixed Hindi is plotted in Figure 3.1a, 3.1b and 3.1b respectively. The test samples are each of the languages plotted in Figures 3.2a, 3.2b and 3.2c. The number of NONE/NOT samples in each of the sub-tasks remain the same since the HOF samples of sub-task A are further annotated for sub-tasks B and C. As apparent from the plots, the samples per class decreases from sub-tasks A to sub-tasks B. This results in a class imbalance in the dataset. Hence to mitigate the imbalance the thesis proposes a simple minority oversampling algorithm named *Interweaving*. The dataset is shuffled once before applying this algorithm. The dataset is not shuffled again. The interweaving algorithm for two labels is given in Algorithm 1. This is done with the intuition that when feeding data batches to the model, the number of samples of each class would remain equal in each epoch. This hypothetically results in the model learning from each class evenly. The dataset is preprocessed using an approach similar to the one applied by Davidson et al. [9].

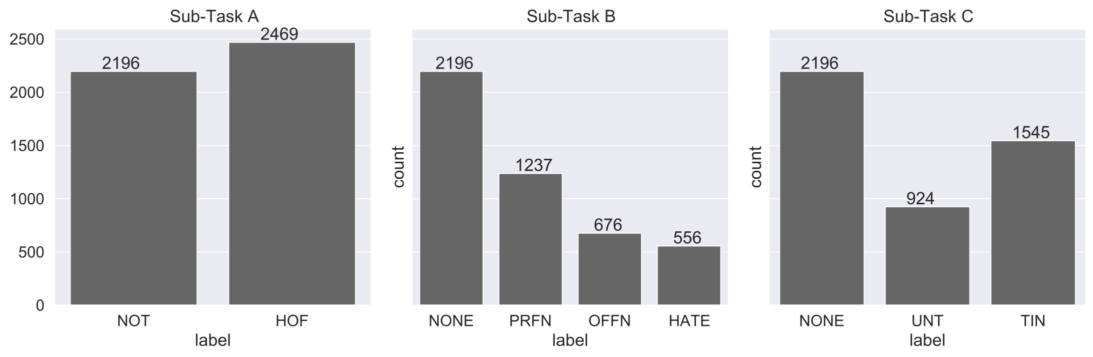
The preprocessing steps are as follows.

1. Hashtags were segmented [1].
   For example *#buildthewall* is segmented into *build the wall*.
2. URLs were removed.
3. Redundant symbols were removed.
4. Redundant whitespaces were removed.
5. The @ symbol from user handles was removed.
6. *RT* prefixes were removed.

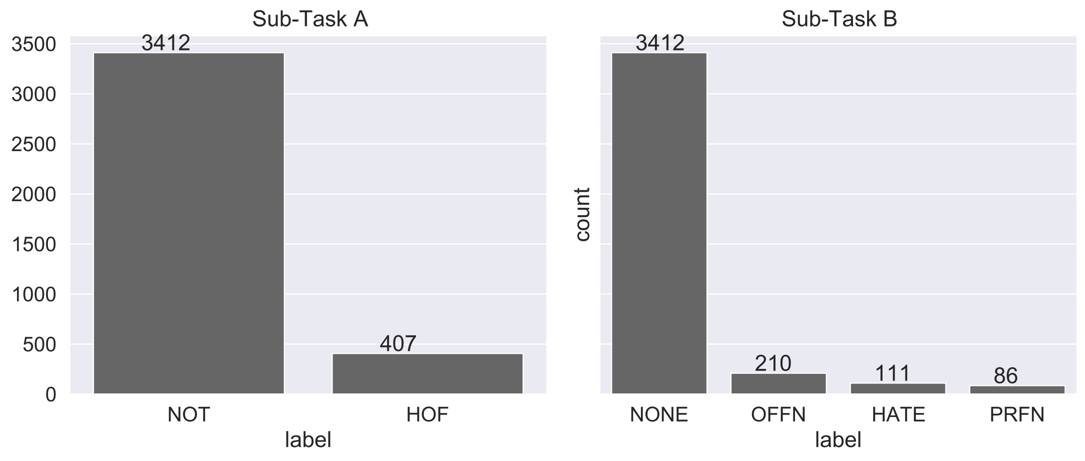
[1]http://www.grantjenks.com/docs/wordsegment/



(a) English

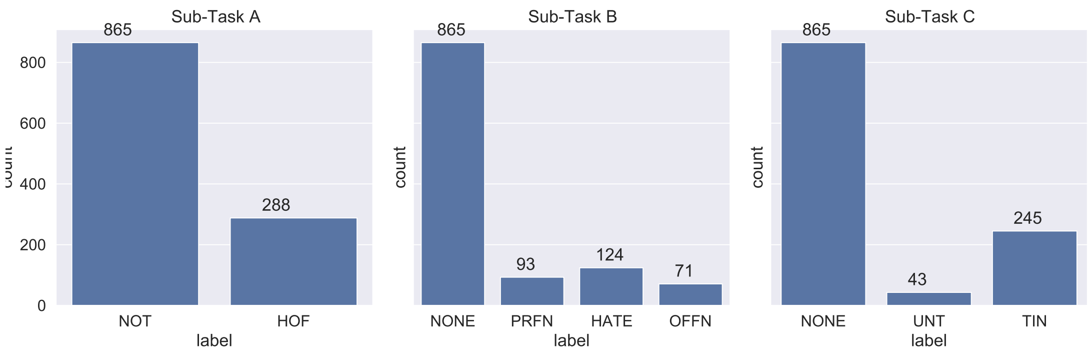


(b) Hindi

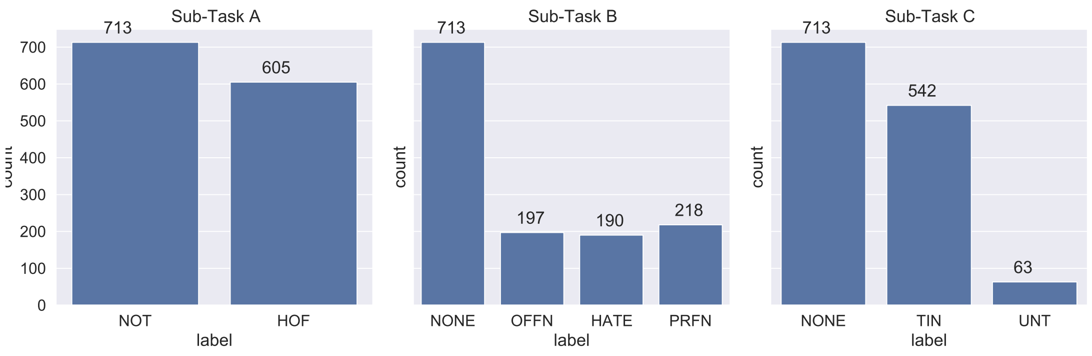


(c) German

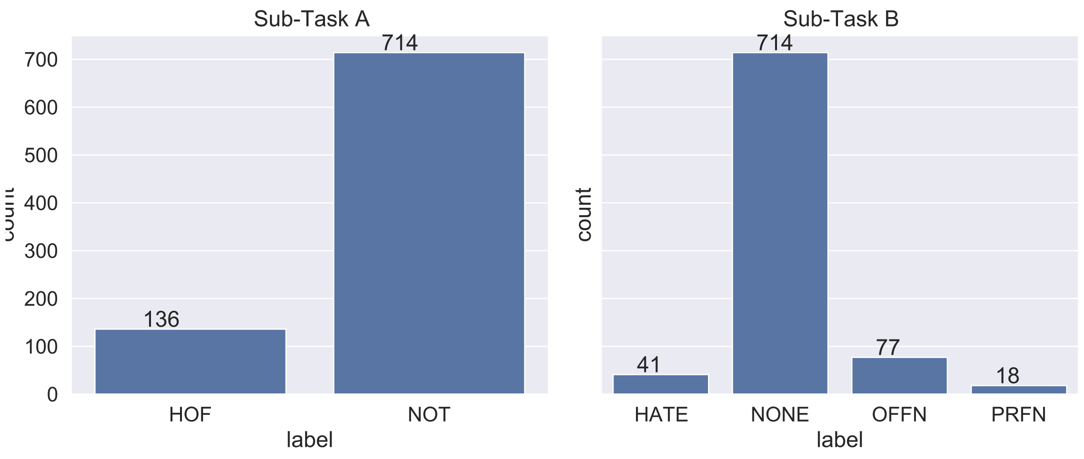
Figure 3.1: Train Set Countplots



(a) English



(b) Hindi



(c) German

Figure 3.2: Train Set Countplots

Algorithm 1: Interweaving

```
Input  : $M \leftarrow$ List of majority samples , $m \leftarrow$ List of minority samples
Output: $B \leftarrow$ Interweaved balanced dataset
1  Initialize $B \leftarrow$ Empty list
2  for $i \leftarrow 1$ to $|M|$ do
3      Append $M[i]$ to $B$
4      if $i >= |m|$ then
5          Initialize $r \leftarrow$ random integer between $1$ to $|m|$
6          Append $m[r]$ to $B$
7      end
8      else
9          Append $m[i]$ to $B$
10     end
11 end
12 return $B$
```

3.3 Methodology

3.3.1 Embeddings

The preprocessed data samples mentioned in the §3.2 are tokenized using NLTK's [20] Tweet-Tokenizer. Each of these tokens are then converted into vector representations using pre-trained embeddings [50]. A word embedding [2] W: $words \rightarrow \mathbb{R}^n$ can be described as parameterized function which maps words in some language to high-dimentional vectors. For example:

$$W(\text{“}fly\text{”}) = (0.11, -0.22, ..., 0.7)$$

W is typically initialized to have some random word vectors for each word which are trained to have meaningful vectors that are useful for some downstream task such as sentiment analysis, classification, etc. In this work, the thesis uses two different types of pre-trained embeddings instead of training them from scratch. If any words in the posts were not found in the embeddings, a zero vector of appropriate dimensions is used. The embedding used for the three languages and their dimensions are given in Table 3.1. The embeddings are briefly described below:

- GloVe [3] [33] is an unsupervised learning algorithm which learns vector representation of words by training on aggregated global word-word co-occurrence statistics from a corpus,

[2]http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/
[3]https://nlp.stanford.edu/projects/glove/

i.e it tries to capture the meaning of one word with the structure of the whole observed corpus.

- FastText [4] [51] is another embedding that takes into account the morphology of the words. It enriches word vectors with sub-word information. Meaning, it can provide embeddings vectors for unknown words. At the time of writing this work, FastText is the only available pre-trained embeddings in Hindi.

Language	*Pretrained Embeddings*	*Dimensions*
English	FastText English	300
Code-Mixed Hindi	FastText Hindi	300
German	GloVe	200

Table 3.1: Embeddings used for each language

3.3.2 Models Used

This section describes the DNN architectures developed for the sub-tasks of the HASOC dataset. All the models have been implemented using the Keras [52] Deep Learning Library. And for other tasks such as evaluating the performance of the models and generating train-validation-test splits, Scikit-Learn [53] is used.

3.3.2.1 Stacked BiLSTM

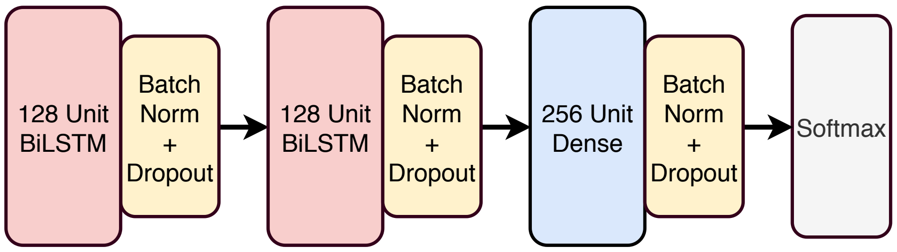


Figure 3.3: Stacked BiLSTM Architecture

The model architecture is given in Figure 3.3. The individual layers and their hyper parameters are given in Table 3.2. The bi-directional outputs of the BiLSTM layers are merged by element-wise multiplication before being fed to the next consecutive layer. Fine-tuning is done

[4]fasttext.cc/

by resetting the weights of the 9th and 10th layer and replacing the final output layer with a new dense layer of appropriate units. All other layers are frozen. The model is trained using Adam optimizer, on its default settings, using a cross-entropy loss function for 20 epochs. Input is fed into the model in batches of 32.

This model is used for all the sub-tasks of the English dataset. It is first trained for classification for sub-task A. Then the model is further fine-tuned for sub-tasks B and C.

No.	**Layer**	**Hyperparams**	**Regularizers** (recurrent, activity, bias)
1.	Input Layer	–	–
2.	Embedding Layer	–	–
3.	BiLSTM (Returns Sequences)	128 ReLu Units	1e-3, None, None
4.	Batch Normalisation	Default	–
5.	Dropout	Rate 0.5	–
6.	BiLSTM	128 ReLu Units	1e-3, None, None
7.	Batch Normalisation	Default	–
8.	Dropout	Rate 0.2	–
9.	Dense	256 ReLu Units	–
10.	Batch Normalization	Default	–
11.	Dropout	Rate 0.5	–
12.	Dense	Softmax Units	–

Table 3.2: Stacked BiLSTM Architecture

3.3.2.2 CNN

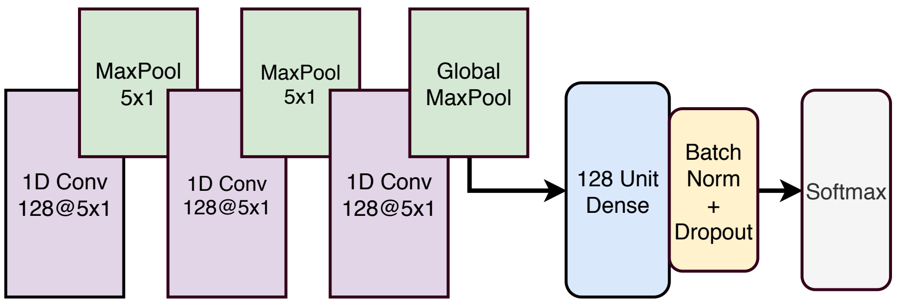


Figure 3.4: CNN Architecture

The architecture of the CNN-based classifier is given in Figure 3.4. It's layers and hyper-parameters are given in Table 3.3. This model was used for all sub-tasks of the German and code-mixed Hindi dataset. Similar to the Stacked BiLSTM, this model is first trained for sub-task A's classification and then fine-tuned for sub-tasks B, C. For fine-tuning, first, the final dense layer is replaced by a new dense layer having c units, where c is the number of classes to classify. Then all the layers were retrained using a small learning rate of $5e-4$ which is decayed by $5e-6$ in every epoch. Here, the model is optimized using *rmsprop* coupled with a cross-entropy loss function. The batch size used is 32 and trained for 20 epochs.

No.	**Layer**	**Hyperparams**	**Regularizers**
			(kernel, activity, bias)
1.	Input Layer	–	–
2.	Embedding Layer	–	–
3.	1D Convolution	128 Filters of size 5	1e-3,1e-3,1e-3
4.	1D MaxPooling	Pool size 5	–
5.	1D Convolution	128 Filters of size 5	1e-3,1e-3,1e-3
6.	GlobalMaxPooling	–	–
7.	Dense	128 Relu Units	–
8.	Batch Normalisation	Default	–
9.	Dropout	Rate 0.5	–
10.	Dense	Softmax Units	–

Table 3.3: CNN Architecture

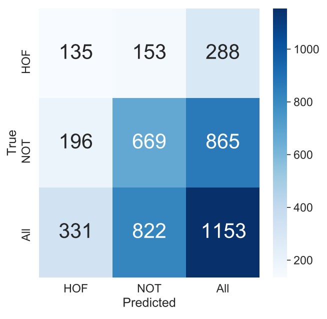
3.4 Results

The metric of evaluation for the HASOC dataset is *Macro-f1*. The results on the test set for each sub-tasks are given in Table 3.4. The rank column of this table provides a comparison of the proposed models to all the models tested by other researchers so far on this dataset [54]. For the English dataset, the BiLSTM model performed quite poorly achieving a macro-f1 score of only 0.61. The model became biased towards the NOT class as is apparent from its confusion matrix shown in Table 3.5a. On further fine-tuning the model for sub-tasks B and C, its performance degraded even further scoring macro f1 of 0.28 and 0.36 respectively. The CNN model used for the Hindi sub-tasks got a macro-f1 score of 0.76 which is around 0.05 shy from the best performing model. This model also lost a lot of performance on fine-tuning for sub-tasks B and C, scoring macro-f1 of 0.26 and 0.58. Both models suffered from false positives and false negatives. In the sub-tasks for the German language, the CNN model achieved a *macro average f1* of 0.52, while the best-achieved score is 0.61.The model was able to classify only 20 samples as HOF. Oversampling by interweaving did not prove to be an effective method for balancing the dataset, especially when the imbalance is large. In sub-task B, the models performed poorly in classifying the OFFEN and PRFN classes. In the German dataset, it completely failed to classify the PRFN class and misclassified most of the OFFEN samples.

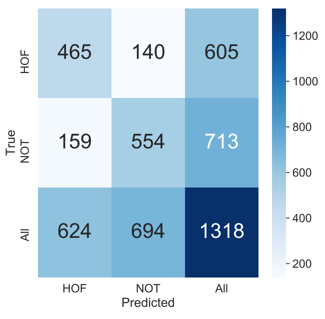
	Model	Sub-*Task*	*Macro-f1*	*Best*	*Rank*
English	Stacked BiLSTM	A	0.6147	0.8396	48
		B	0.2821	0.5446	31
		C	0.3619	0.5111	28
Hindi	CNN	A	0.756	0.8149	20
		B	0.2583	0.5812	35
		C	0.3402	0.5754	19
German	CNN	A	0.5223	0.6162	11
		B	0.268	0.3468	7

Table 3.4: Official Results

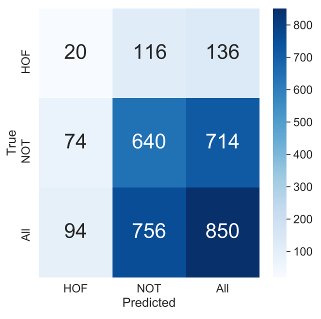
3.5 Error Analysis



(**a**) English



(**b**) Hindi



(**c**) German

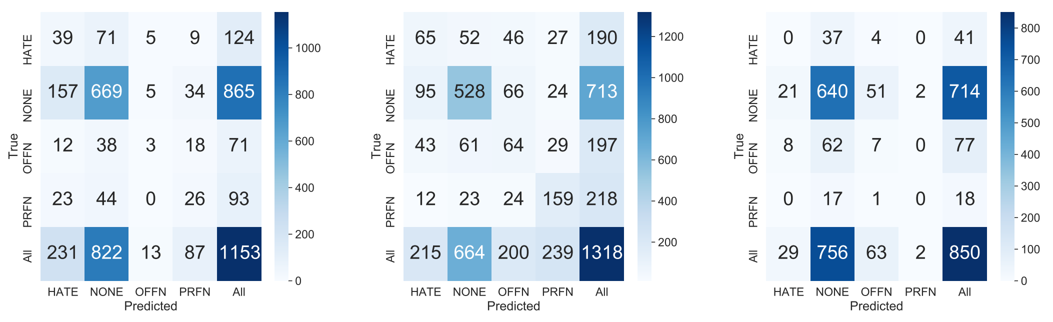
Figure 3.5: Confusion Matrices For Sub-Task A

The English test-set dataset contains code-mixed samples which the Stacked BiLSTM did not classify correctly as HOF. For example:

- *After the* भाषण *on the fear, hatred for #minorities by the #ModiRajya chief, trolls, bullying* *of #Muslims have not, repeat not stopped on social media. Old habits die hard They will not change.* #कुत्ते की दुम है सब के सब...

- *It wont affect to alia but it will effect to your TRP...* भोसडीके

This is because the model is trained using English embeddings and in the above-mentioned examples, but the hate and offence are within the Hindi parts. Furthermore, in the first example, the insult '#कुत्ते की दुम है सब के सब... 'is of complex nature where the offender is comparing people to dogs. This is derived from a different Hindi insult, which means *some people never change just like a dog's tail can never be straight*. The models in this study are not trained on such contextual information, which further contributes to the misclassification. BiLSTM model also failed to classify some obvious examples in all the sub-tasks. This is probably due to underfitting the dataset. This problem with complex samples is also found in the German and Hindi Models. Although, in the German sub-tasks, the problem might be attributed to its unbalanced dataset. Its also worth nothing that NLTK's TweetTokenizer is not built for the Hindi language, and performs character-level tokenization on the Hindi words.

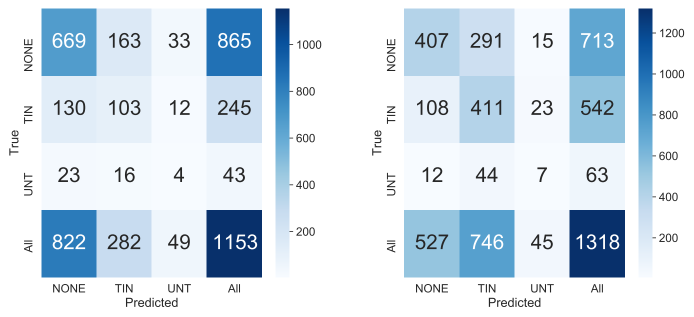


(a) English **(b)** Hindi **(c)** German

Figure 3.6: Confusion Matrices For Sub-Task B

Since all three tasks are interrelated, errors made in sub-task A resulted in misclassification in sub-task B and sub-task B as well. Further error analysis found that code-mixed samples were axlso misclassified in Hindi sub-task B, C. For example:

- *Ch****i ke alawa bhi 10 cheez hai. Jab dekhi f*** f*** karti reh*
- *Ye hain 'Bankey Bihari' , ye 19 July ko hi bhartee huay hain TROLL ARMY mein .... Inki bhasha dekhiye , Maa baap se bhi ye shayad aise hi baat karta hoga .. Kaash ye shakha ki jaga school gaya hota ! May Maa Durga ease his pain of not achieving anything except TROLLSHIP in life !*



(a) English **(b)** Hindi

Figure 3.7: Confusion Matrices For Sub-Task C

3.6 Discussion and Conclusion

A lot remains to be explored in the field of hate speech classification in languages such as Hindi and German. The best performing models in each of the languages have macro-f1 scores below $\%90$. This drops down further for sub-tasks B and C, where the best score is just 0.3468 for German sub-task C. In this chapter, the thesis contributed two different DNN based approaches, Stacked BiLSTM and CNN, for tackling the classification of hate speech in Indo European languages. The performance of both of these models leaves much to be desired. The simple minority oversampling method used to address imbalance did not work as hypothesized. It was found that code-mixing is another challenge that must be faced while dealing with social media posts, especially in low resource languages such as Hindi, where people instead tend to use Hinglish a lot. Such code-switching between Hindi and English is also found in the English dataset. This necessitates the use of multi-lingual approaches instead of the mono-lingual ones used in this study. In future work, it will be interesting to experiment with pre-trained multi-lingual models such as M-Bert[5] and XLM-RoBERTa[55].

This work as been published [56] in proceeding of HASOC 2019 at FIRE'19, Kolkata. The code for the models used in this study are publicly shared online[6].

[5]https://github.com/google-research/bert

[6]https://github.com/cozek/hasoc-2019-falsepostive

Chapter 4

Checkpoint Ensemble of Transformers for Hate Speech Detection

4.1 Introduction

It is difficult to curb the spread of offensive content in social media. Miscreants take advantage of the wall of anonymity provided by online communication methods to spread offensive content. Online social media platforms, communities and technology companies have invested heavily in technologies to help cope up with this abusive behaviour. So far, the most effective way to tackle this problem is through automated computational techniques such as the ones covered in Chapter 2 of this thesis. But, the ever-changing nature of this problem (§1.2) means that there is no single solution to this problem.

This chapter is a continuation of the mechanisms described in Chapter 3. This chapter proposes three transformer-based ensemble classifiers for the task of automated detection of hate-speech. The classifiers are evaluated on the newly proposed massive-dataset of OffensEval-2020[1] [57].

The rest of the chapter is organized as follows. The dataset and classification tasks are described in §4.2. The Ensembling methodology and algorithm used for building the classifiers is explained in §4.3. The individual classifiers used in each task is described in §4.5. The performance and error of the classifiers are analyzed in §4.6. And finally, we conclude the chapter in §4.7.

[1]https://sites.google.com/site/offensevalsharedtask/home

4.2 OffensEval-2020 Dataset and Classification Sub-Tasks

The massive dataset of OffensEval-2020 is provided in 5 different languages, namely Arabic, Danish, English, Greek and Turkish. This chapter makes use of the English subsection of this multilingual dataset. Each sample of the dataset is *loosely annotated* for 3 different classification sub-tasks. This means that each sample is associated with an *Average Confidence* measure and it's corresponding *Standard Deviation*. This *Average Confidence* is the average of the confidences with which a collection of supervised models predicted a sample as belonging to the positive class. In this chapter, these *loose annotations* are converted into *hard annotations* by assigning a sample to the positive class if its average confidence for that class is greater than or equal to 0.50. The positive class for each Sub-task is marked with $\diamondsuit$ symbol in the list 4.2 below. However, Sub-task C has no positive class, instead, the average confidence of each class is given. The taxonomy of the classification sub-tasks are given below, *verbatim* as given by the dataset authors. The training and testing splits of this dataset is given in Figure 4.1 and Figure 4.2 respectively.

OffensEval-2020 Classification Sub-tasks

A) *Offensive language identification*

- Not Offensive (NOT) : A post with no offensive language or profanity.
- Offensive$^\diamondsuit$ (OFF) : A post containing offensive language or a targeted (veiled or direct) offense.

B) *Automatic categorization of offense types*

- Targeted Insult and Threats (TIN) : A post containing an insult or a threat to an individual, a group, or others (see categories in sub-task C).
- Untargeted$^\diamondsuit$ (UNT) : A post containing non-targeted profanity and swearing.

Posts containing general profanity are not targeted, but they contain non-acceptable language.

C) *Offense target identification*

- Individual (IND) : The target of the offensive post is an individual: a famous person, a named individual, or an unnamed person interacting in the conversation.
- Group (GRP) : The target of the offensive post is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else.

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

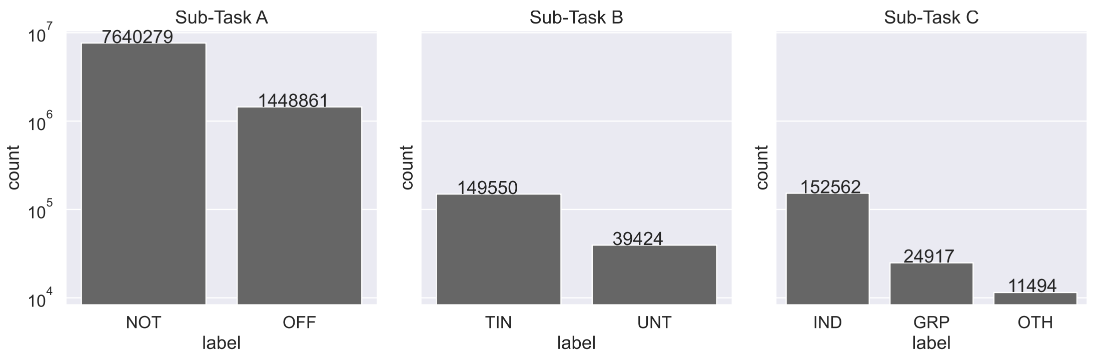


Figure 4.1: OffensEval-2020 English Dataset

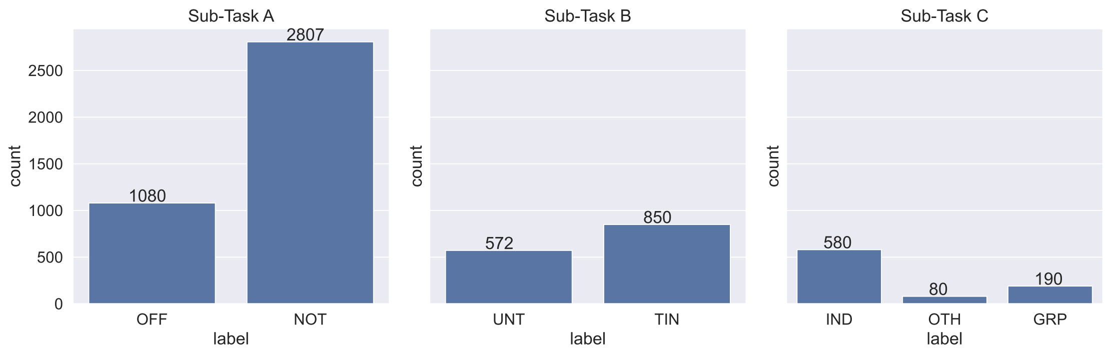
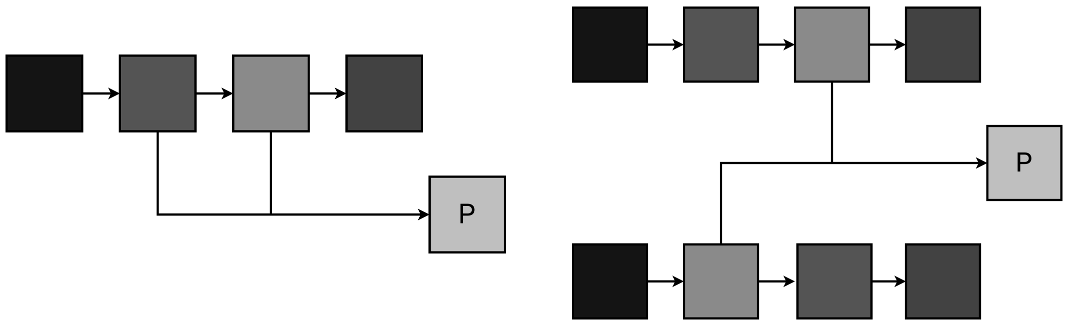


Figure 4.2: OffensEval-2020 English Test Dataset

4.3 Checkpoint Ensembling

Model Ensembling refers to the method of constructing a single classifier from a collection of different classifiers [58]. However, creating ensembles of large DNNs is expensive and often infeasible. It is generally not possible to train multiple models with limited time and GPU resources. Hence, based on the work by Chen et al. [1], the thesis proposes a simple *Checkpoint Ensembling* method for creating ensembles of large pre-trained models.

In *Checkpoint Ensembling*, a copy of the model is saved at each checkpoint. These copies are later combined in some fashion to make the classification. In this thesis, the dev-set predictions and weights of the DNN model are saved at each epoch. The dev-set predictions are then used to determine which checkpoints of the model to use for the ensemble using Algorithm 3. Algorithm 3 describes selecting a model if using its predictions improve the selected performance metric, otherwise not. For the OffensEval sub-tasks, the performance metric is *Macro-f1*. Thereby, a list of models with which to create the ensemble in determined. Algorithm 3 is called twice with $reverse$ set to $True$ and then $False$. If the ensemble doesn't improve the metric, simply the best model found during training is chosen. After determining the models, Algorithm 2 is used to make the predictions. In Algorithm 2, the predictions of the final classifying layer of the chosen models are added and $argmax$-ed along each row to obtain the final prediction. One variation of this would be adding and averaging as described in the original work [1], however experimentation revealed that the performance is better with the $argmax$ approach. This suggests that the suitable approach might vary with use case.



(a) Checkpoint Ensemble **(b)** Traditional Ensemble

Figure 4.3: Diagram based on [1]. The square boxes are models at each step of a training process. Lighter shade means better performance. In traditional ensemble (4.3b), we pick the best model found during the training process. In checkpoint ensemble (4.3a), we pick the different checkpoints of the same model.

Algorithm 2: Make Prediction

```
Input  : $m \leftarrow$ model ids chosen for ensemble
Output: $preds$
begin
    Initialize: $E[N][C] \leftarrow$ Zero Matrix
    for $i$ in $m$ do
        Load model with weights at epoch $i$
        $p \leftarrow model.predict(samples)$
        $E \leftarrow E + p$
    end
    $preds \leftarrow$ Index of max value across each row of $E$
    return preds
end
```

Algorithm 3: Naive Checkpoint Ensemble

```
Input  : $A \leftarrow$ True labels, $reverse \leftarrow boolean$
         $P \leftarrow$ Model predictions at each epoch
         $N \leftarrow$ Number of samples
         $C \leftarrow$ Number of classes
         $reverse \leftarrow boolean$
         $models \leftarrow \{\}, val \leftarrow 0$
Output: $models$ , $val$

Initialize $Z[N][C] \leftarrow$ Zero Matrix
Initialize $\epsilon \leftarrow len(P)$ // Number of Epochs
begin
    if $reverse$ then
        $range \leftarrow \epsilon$ to 0
    end
    else
        $range \leftarrow$ 0 to $\epsilon$
    end
    for ($e \leftarrow range$) do
        $temp \leftarrow Z$
        $temp \leftarrow temp + P[e]$
        if $metric(A, temp) > val$ then
            $Z \leftarrow Z + P$
            $models \leftarrow models \cup e$
            $val \leftarrow metric(A, temp)$
        end
        else
            continue
        end
    end
    return $models, val$
end
```

4.4 Transformer Models

Vaswani et al. (2017) [11] demonstrated that a neural network built solely using attention mechanism could outperform other networks relying on recurrence or convolutions. They called this architecture, the Transformer. These transformers were not only more parallelizable but also trained significantly faster. Based on this architecture, Devlin et al. (2018)[12], introduced *Bi-Directional Encoder Representations from Transformers* (BERT), which is built using multiple layers of bi-directional transformers. Being trained on very large bodies of text in an unsupervised manner, BERT models have bi-directional context rich representations of words. This allows for BERT models to be task-agnostic, i.e, be easily fine-tuned for various tasks such as text classification, question answering, etc. Currently BERT based models hold the highest ranks in GLUE benchmark [59] for NLP. The classifiers developed by this chapter uses some of the more recent BERT based models.

4.5 Classifiers

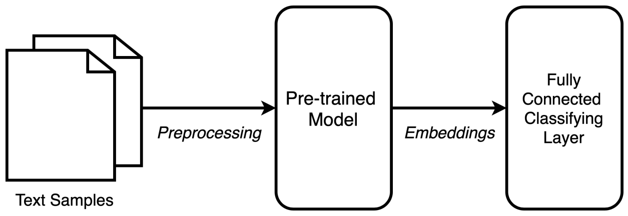


Figure 4.4: Transfer Learning Model

The classifier models built for each of the Sub-tasks are described in this section. All the classifiers use a typical *transfer-learning* approach (see Figure 4.4) coupled with *Checkpoint Ensembling* (§4.3). The preprocessing steps simply include adding special tokens and padding the shorter samples to the max length used for each classifier. These special tokens include transformer specific delimiters between sentences and padding tokens. For adding special tokens between sentences we used NLTK's [20] Punkt sentence tokenizer. And as for writing the code for the models and training scripts, PyTorch[2] [60] was used. The thesis makes use of the HuggingFace Transformers Library[3] for all the pre-trained models. The data-splits used for training and validation are such that the percentage of samples for each class is the same for all splits. The random seed is set to 42 wherever applicable.

[2]https://pytorch.org/

[3]https://github.com/huggingface/transformers

4.5.1 Sub-Task A: Ensembled GPT-2

The classifier developed for Sub-Task A is built on top of GPT-2 which is a large transformer-based [11] language-model proposed by Radford et al. [61]. GPT-2 trained on 40GB of internet text and has many capabilities, the main being able to generate synthetic text. It can generate text of such high quality that it could easily be mistaken for being written by an actual human.

But, due to the extremely large-size of GPT-2 (1.5 billion parameters), the thesis instead makes use of a distilled version of the same, called DistilGPT2. Distil* [62] is a class of compressed transformer models that has faster training and inference time while being small in size. Sanh et al. show that Distil* retains up to 97% performance of the original models. These compressed models made it possible for the thesis to work on the OffensEval-2020's massive datasets with high-performance base models with limited resources and time.

The chapter uses 70% of the data for training and 0.001% (13633 samples) for validation. This choice in size was made due to time constraints. For training the classifier, first, each text sample is converted into a sentence-matrix by extracting $786 - dimensional$ word embeddings from pre-trained DistilGPT2. These are then fed into a dense layer having c units which makes the classification. Here c is the number of classes. The entire model is fine-tuned using a cross-entropy loss-function with a small learning rate of $1e - 4$ for 5 epochs and coupled with *Checkpoint Ensembling* as described in §4.3. For optimizing the model, we use Ranger which is a combination of two optimizers, RAdam [63] wrapped with LookAhead [64]. The (k, α) parameters of the optimizer is set to (5, 0.5). The batch-size used is 64 and maximum-sequence length is set to 399. Checkpoint ensembling improved the dev-set macro-f1 score from 0.9656 to 0.9663.

4.5.2 Sub-Task B: Ensembled RoBERTa

Lie et. al. [65] identified the short-comings of BERT [12] and introduced RoBERTa, a robustly optimized version of BERT. Compared to the original, RoBERTa is trained on a larger dataset, uses dynamically masking out tokens instead static masking, etc. It's pre-trained version, called *Roberta Sequence Classifier* by Wolf et al. [66] is used for sub-task B. The classifier is trained for 20 epochs with early stopping patience set to 4. We use 85% of the data as train set and the rest as dev set. The maximum sequence length is set to 245 and batch size used is 128. The other parameters and hyper-parameters are the same as that of Sub-Task A (§4.5.1). Checkpoint ensembling improved macro-f1 on dev-set from 0.8881 to 0.8907.

4.5.3 Sub-Task C: Ensembled DistilRoBERTa with Attention Mask Dropout

This classifier is built using the pre-trained distilled version of *Roberta Sequence Classifier* by [66], coupled with a slightly modified *Checkpoint Ensembling*. Here, instead of directly using the dev set predictions, first, it is sorted in decreasing order of their macro-f1 scores. Also, the thesis experiments with the application of dropout to the attention masks which are generally used within transformer models. Attention Masks specify which of tokens of the sentence the model should attend to. In *Attention Mask Dropout*, with probability p, $d\%$ of attention masks of the tokens in the sentence are randomly dropped, i.e, set to 0. In this work, (p, d) is set to (0.50, 30%) respectively. Attention mask dropout is used in an attempt to mitigate the poor grammar which is often encountered in social media texts (1.2). Similar to the original dropout technique [67], it is only used during training. The maximum sequence length is set to 245. The pre-trained model is fine-tuned using a cross-entropy loss function for 20 epochs with early stopping patience set to 4. We use a batch-size of 120 and learning rate of $1e-04$ with Ranger optimizer. Like in previous classifiers, the (k, α) parameters of the optimizer are set to (5, 0.5). Here 95% of the labelled data is used for training and the rest as dev set. Ensembling improved the dev-set macro-f1 from 0.8148 to 0.8281.

4.6 Results and Error Analysis

		Macro-f1			
	Model	*Dev Set*	*Test Set*	*Best*	*Rank*
Task A	Ensembled DistilGPT2	0.9663	0.90989	0.92226	25
Task B	Ensembled RoBERTa	0.8907	0.55181	0.74618	31
Task C	Ensembled DistilRoBERTa	0.8281	0.6168	0.7145	15

Table 4.1: Official Test Set Results

The dev and test set results for each of the classifiers are given in Table 4.1. Similar to the results in the previous Chapter 3, the rank column of this table provides a comparison of performance of this chapter's approach versus the approaches used by other researchers [57]. The test set contained 3887, 1422, 850 text samples for subtasks A, B and C respectively. Ensembled DistilGPT2 is able to cross the 0.90 mark and is just shy of 0.2 from the best. Perhaps using the full dataset instead of just 70% might have closed the gap. This shows the need for techniques like *Model Quantization*[4] and *Weight Pruning*[5] to deal with massive datasets such as the ones used in this chapter (§4.2). Ensembled RoBERTa and Ensembled DistilRoBERTa performed quite poorly. They clearly overfit the data as the difference between the dev set and test set f1 scores is quite large. Later experimentation revealed that attention mask dropout in sub-task C actually hurt the performance of the model. Without it, the dev-set macro-f1 is 0.8275 and sorted checkpoint ensembling improves the score to 0.8336. Ensembled DistilGPT2 for Subtask A is

		Macro		
	Model	*Precision*	*Recall*	*f1*
Task A	Ensembled DistilGPT2	0.89046	0.94546	0.90989
Task B	Ensembled RoBERTa	0.72424	0.58696	0.55181
Task C	Ensembled DistilRoBERTa	0.72362	0.58820	0.61680

Table 4.2: Detailed Test Set Performance

able to correctly classify every offensive sample, except for two which were predicted as not offensive. Those two samples being

1. *@USER @USER Dehumanize? He barely has a reflection of human*
2. *@USER @USER 'Respect the result' is a mendacious soundbite trotted out by charlatans daily.*

This is perhaps due to the complex use of language, use of rare words such as *mendacious*, *trotted* and absence of typical profane words. As seen from the confusion matrix in Figure 4.5a, this classifier has a bias towards the offensive class.

Ensembled Roberta for Subtask B performed poorly and became biased towards the targeted class. It failed to distinguish properly between targetted and untargeted samples. Untargeted samples such as

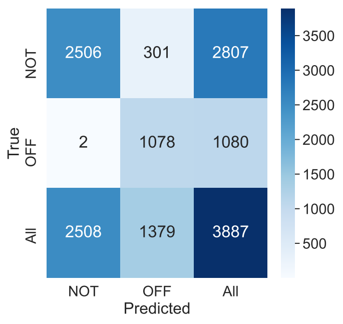
- "*@USER Yeah my deck was insane that run, but normally I suck lol* "

were predicted as targeted. We found that samples which contained the "*@USER*" token were mostly mistaken as targeted. The confusion matrix is given in Figure 4.5b.

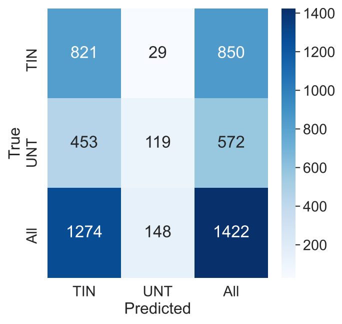
Ensembled DistilRoberta for Subtask C misclassified most of the samples of 'other' class as targeting individuals, as apparent from its confusion matrix in Figure 4.5c. The same effect can be seen in the group class. Also, we found that this classifier made many mistakes where emojis

[4]Quantization means decreasing the numerical precision of a model's weights

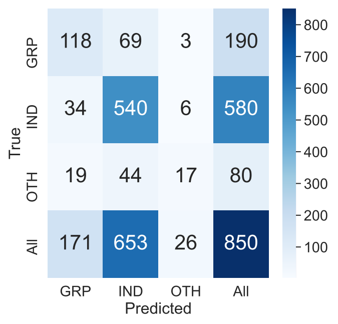
[5]Pruning removes parts of a model to make it smaller and faster.



(a) Task A Test Set Confmat



(b) Task B Test Set Confmat



(c) Task C Test Set Confmat

Figure 4.5: Confusion Matrices

are used. For example, ●"*I'm the only* 👆 *to get the job done ..... ion kno a nigga dat can cover for me*" is misclassified as belonging to the individual class rather than group targeted. In this example, the 👆 is the "Index Pointing Up" emoji which means "me" in the current context. The model is unable to get the context from the emojis.

4.7 Discussion and Conclusion

In this chapter, the thesis built three different transformer-based classifiers for the three subtasks of OffensEval 2020. The thesis experimented with creating ensembles using Checkpoint Ensembling. Although checkpoint ensembling improved the performance of the models, the improvements are quite small. But, considering how cheap and simple the method is, it can't be dismissed completely. Further work is required to find out better ways to utilize this particular method of ensembling. Testing out other ensembling techniques such as Meta Classifier by Malmasi et al [68] is left as future work. The thesis failed to utilize the full dataset for training the models due to time and resource constraints. Albeit Distil* models were used, further optimization using techniques such as Model Quantization is could have been performed. Experiments found that attention mask dropout did not work as expected. We think this needs further work and experimentation with different values of the p and d hyper-parameters.

The code[6] for the work done in this chapter is available as Colab[7] Notebooks for reference of the reader.

[6] https://github.com/cozek/OffensEval2020-code
[7] http://colab.research.google.com

Chapter 5

Automated Aggression Identification using Transformers

This chapter deals with the identification of other types of content in social media that has the same negative effect of causing harm to society. Particularly, this chapter focus on the detection of online aggression, trolling and cyber-bullying in social media texts. Studies find that incidents of online abuse can cause a host of negative impacts ranging from mental and psychological health problems, deactivation of accounts to instances of self-harm and suicide. Just like hate speech, the rapid growth of user-generated content [69] has made traditional human flagging methods infeasible for moderating the aforementioned content. Although similar to hate speech, the most important distinction is the target. Hate-speech is usually targeted at groups, whereas cyber-bullying is targeted towards individuals. Also, aggression is often veiled using clever use of language.

The work in this chapter has been done in parallel with Chapter 4, hence much of the methodologies are the same. But, here the thesis proposes a *Attention Head* classification layer (§5.2.1) to use in conjunction with pre-trained models. This chapter uses the TRAC-2 Aggression Identiciation Dataset [46] which is annotated for the aggression classification tasks and features three classification sub-tasks (§5.1). The models built for these tasks are described in §5.2.2. Their performance and shortcomings are discussed in §5.3. Finally, the chapter in concluded in §5.4.

5.1 TRAC-2 Dataset and Classification Sub-Tasks

Similar to the dataset in Chapter 3, the TRAC-2 Dataset is given in 3 different languages, namely Hindi, Indian Bengali and English. It is compiled from randomly sampled and annotated YouTube comments. This dataset features 2 classification sub-tasks in each of the languages.

			Word Count			
Dataset	*Split*	*Size*	*Mean*	*Max*	*Std*	*Samples below 50 words*
English	Train	4263	17.23	806	33.47	93.31%
	Dev	1066	17.52	457	30.25	93.34%
	Test	1200	39.68	1390	82.36	77.41%
Hindi	Train	3984	17.45	557	25.38	95.41%
	Dev	997	17.18	230	20.62	93.98%
	Test	1200	23.48	669	38.19	89.92%
Indian Bengali	Train	3826	9.08	154	11.17	98.64%
	Dev	957	8.66	182	12.29	98.64%
	Test	1188	7.41	113	9.13	99.24%

Table 5.1: TRAC-2 Dataset Statistics

These are defined as follows.

TRAC-2 Classification Sub-Tasks

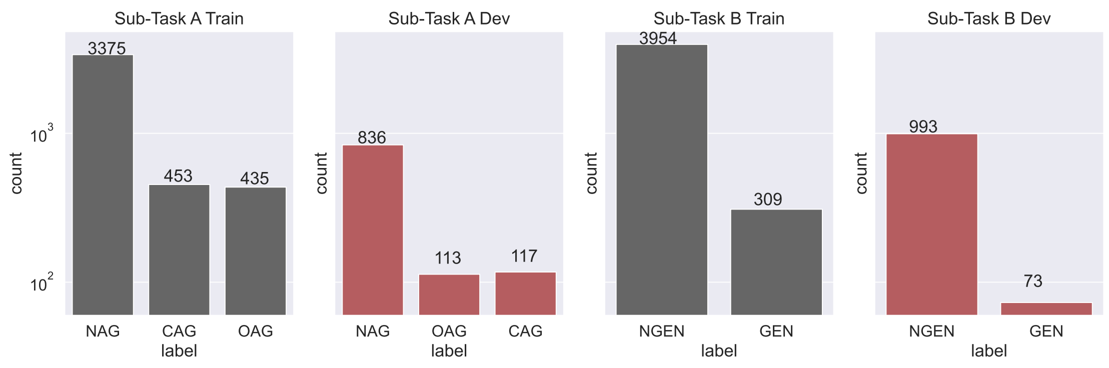
A) *Aggression Identification*

- Covertly Aggressive (CAG): Indirect attack often packaged as (insincere) polite expressions.
- Overtly Aggressive (OAG): Direct with usage of specific lexical/syntactic structures.
- Non-aggressive (NAG): Neither of above.

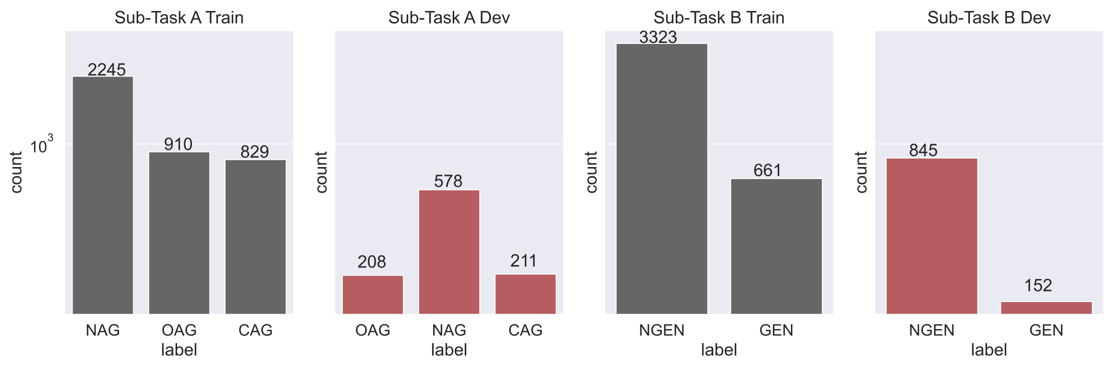
B) *Misogynistic Aggression Identification*

- Gendered (GEN): Comment in which verbal aggression is aimed towards
  - The stereotypical gender roles of the victim as well as the aggressor.
  - Aggressive reference to one's sexuality and sexual orientation, including homophobia.
- Non-gendered (NGEN): Comments which are not gendered.

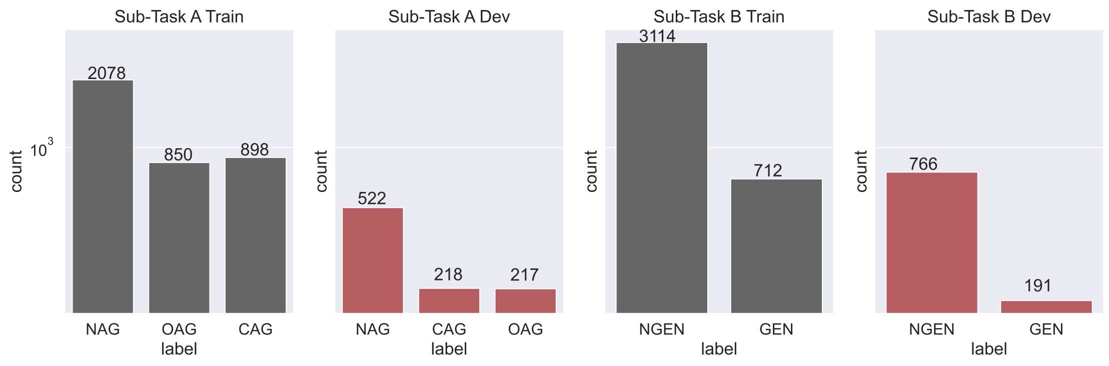
As can be seen the figures, the dataset is imbalanced with NAG (for subtask A) and NGEN (for subtask B) occurring more frequently in all the three languages. The NGEN category occurred as high as 93.15% in the English development dataset. Even though it is unbalanced, this is in line with the proportion of aggressive and non-aggressive comments in real-life as has been mentioned in Gao et al. [70], with the only exception being the Hindi test dataset.



(a) English Train and Dev Set Countplot



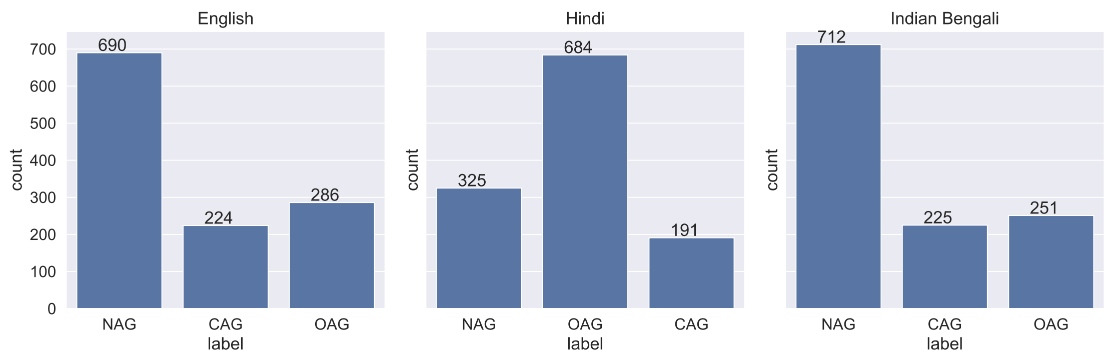
(b) Hindi Train and Dev Set Countplot



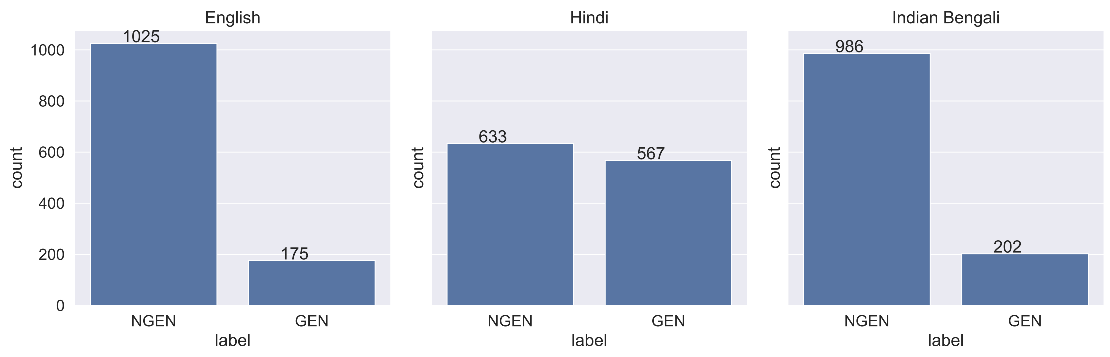
(c) Indian Bengali Train and Dev Set Countplot

Figure 5.1: Train Set Countplots

Further exploration reveals that the length of the samples, i.e. the number of words, vary widely (see Table 5.1). The minimum number of words in samples of each split is 1, while the maximum could go up to 1390 words. However, most of the samples have lengths below 50 words. Since long text inputs are not supported by the classifiers used in this study, they were split into multiple samples, each containing a max of 200 words. The splits were given the same label as one it was made from. The head 50 words of each consecutive split contained the trailing



(a) Sub-Task A Test Set Countplot



(b) Sub-Task B Test Set Countplot

Figure 5.2: Test Set Countplots

50 words of its previous split.

It also interesting to that all three of the datasets, including English, are *code-mixed*. The degree of code-mixing is found to be higher in the Hindi and Indian Bengali datasets. For example, the English dataset contains comments in which English alphabets are used to express Bengali or Hindi dialects.

Code-Mixing Examples

- "*haste haste pet fete gelo*" which translates to "*laughed so much by stomach exploded*" in Bengali.
- "*we r love this vlog bhai*", here the Hindi word "bhai" means brother in English.

5.2 Methodology

Similar to the classifiers built in Chapter 4, the thesis builds three DNN based classifiers using the PyTorch deep learning framework and the HuggingFace Transformer's Library. The text preprocessing steps are also the same. The classifiers used for the sub-tasks in this chapter makes use of a similar transfer-learning approach. Here, the thesis uses an *Attention Head Layer* instead of a linear layer as the classification layer on top of a pre-trained model. Checkpoint Ensemble is also used in some of the classifiers.

The Attention Head layer is described in §5.2.1. Each of the different classifiers built for each language is explained in §5.2.2.

5.2.1 Attention Head Layer

The attention head is based on the work done by Zhou et. al (2016) [71] with a few modifications. In the original work, the attention head uses features extracted from BiLSTM layers. Here, in place of BiLSTM, the thesis uses embeddings extracted from a pre-trained transformer. The original work uses $tanh$ as the activation function, the thesis instead uses $penalized - tanh$, which is demonstrated to work better for NLP tasks [72]. This is combined with a cross-entropy loss function. Also, the thesis does not apply $softmax$ on the output of the layer and instead directly uses $argmax$ make the prediction. The mechanism of the layer is described as follows:

Let H be a matrix consisting of embedding vectors $[h_1, h_2, .., h_T]$ where T is the sentence length.

$$M = penalized\tanh(H) \tag{5.1}$$

$$\alpha = softmax(w^T M) \tag{5.2}$$

$$r = H\alpha^T \tag{5.3}$$

Where $penalized\tanh$ is defined as:

$$penalized\tanh = f(x)\begin{cases}\tanh(x) & x > 0,\\ 0.25\tanh(x) & x \leq 0\end{cases} \tag{5.4}$$

And $H \in R^{d^w \times T}$, d^w is the dimension of the embedding vectors, w is a trained parameter vector and w^T is a transpose. The dimension of w, α, r is d^w, T, d^w respectively. The final sentence-pair representation used for classification is obtained from:

$$h^* = penalized\tanh(r) \tag{5.5}$$

During implementation, the layer's weights are initialized using the popular Xavier Uniform method [73].

5.2.2 Classifiers

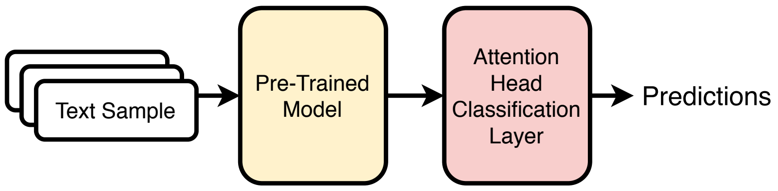


Figure 5.3: Classifier Architecture

This section described the classifiers used for each of the languages. The basic architecture of the classifiers is given in Figure 5.3. *Checkpoint Ensembling* (§4.3) is used wherever it resulted in increased performance.

5.2.2.1 English: RoBERTa and DistilRoBERTa

The pre-trained models used in the classifiers for English sub-task A and B are RoBERTa and DistilRoBERTa respectively. These pre-trained models have been described already described in §4.5.2. The classifiers couples the pre-trained model with an attention head layer which makes the classification. Similar to the classifiers in §4.5, the entire architecture is fine-tuned using the Ranger optimizer and a cross-entropy loss function. The learning rate for both of the classifiers is set to $1e-4$. The (k, α) hyper-parameter of the optimizer are set to $(5, 0.5)$ and $(6, 0.5)$ with a weight decay of $1e-5$ for sub-tasks A and B respectively. The training is run for 20 epochs with early-stopping patience set to 4. *Checkpoint Ensembling* is applied for English sub-task A which increased the weighted-f1 on the dev-set from 0.8070 to 0.8124.

5.2.2.2 Hindi and Indian Bengali: XLM-RoBERTa

XLM-RoBERTa [55] is a large transformer model based on RoBERTa. It is trained on a colossal multi-lingual dataset of two terabytes and containing up to 100 different languages, including Hindi and Bengali. It has attained state-of-the-art scores on multiple cross-lingual benchmarks at the time of writing this thesis. This model is aimed at tackling the *curse of multi-linguality* problem often run into by cross-lingual models. The thesis uses the pre-trained version of this model in the classifiers for all the sub-tasks in the Hindi and Indian Bengali languages.

This classifier uses the same architecture, optimizer, training epochs, and early stopping as the one used for the English sub-tasks with some differences in hyper-parameters. The optimizer's (k, α) is set to $(6, 0.5)$ with weight–decay of $1e-5$. The batch-size for the Bengali sub-tasks (A,B) is $(22, 24)$ and 32 is both Hindi sub-tasks. *Checkpoint Ensembling* is applied for both the Hindi sub-tasks. This increased the weighted-f1 from 0.7146 to 0.7160 for Hindi sub-task A and from 0.8908 to 0.8969 for Hindi sub-task B. And as for the Bengali sub-tasks, the best model found during training is used as ensembling did not yield any improvements.

5.3 Results and Error Analysis

	Sub-Task	*Weighted-f1*	*Best*
English	A	0.698	0.802
	B	0.862	0.871
Hindi	A	0.793	0.812
	B	0.869	0.878
Indian Bengali	A	0.788	0.821
	B	0.915	0.938

Table 5.2: Official Test Set Results. Hindi Sub-Task B obtained rank 2

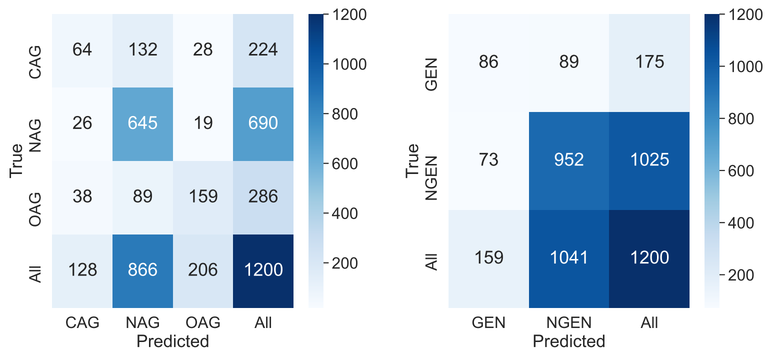
The dev and test set performance for each sub-task in English, Hindi and Indian Bengali are given in Tables 5.3, 5.4 and 5.5 respectively. Model performance is evaluation using *Weighted-f1*, same as the dataset authors. The performance of the classifiers are compared in Table 5.2. In English Sub-Task A, the model underfits the dataset, scoring of 0.698. The classifier for English Sub-Task B did better, scoring 0.862, which is close to the best model's score of 0.871. And for the rest of the classifiers, except for Indian Bengali Sub-Task A, the thesis's classifiers perform competitively close of the best classifiers. The classifier for Hindi Sub-Task B got rank 2 amongst all the classifiers built by other researchers for the dataset [74]. The error and classifier performances for each of the languages are analysed individually in the rest of this section.

5.3.1 English

			Macro			
	Model	*Split*	*Precision*	*Recall*	*f1*	*Weighted-f1*
Sub-Task A	RoBERTa	Dev	0.6418	0.5883	0.6106	0.8070
		Test	0.6722	0.5921	0.6130	0.6986
Sub-Task B	DistilRoBERTa	Dev	0.7201	0.6866	0.7016	0.9260
		Test	0.7277	0.7101	0.7183	0.8623

Table 5.3: English Dev and Test Set Results

RoBERTa for English sub-task A unable to correctly classify long samples. It was found that the mean of the number of characters of all the misclassified samples of this model to be $\approx 474.37(std. \approx 699)$, while that of the correctly classified samples is $\approx 130(std. \approx 264.52)$. This could be because the pre-trained models do not support samples that have a large number of tokens and hence is unable to capture the full concept of what is being implied. Misclassifications show that this model had a hard time correctly classifying the covertly aggressive. For example, the following CAG samples were incorrectly classified as NAG.



(a) Sub-Task A **(b)** Sub-Task B

Figure 5.4: English Test Set Confusion Matrices

- *the irony is that these false accusations made on the men are by the high society women who are so called educated because they have clever mind and they make this false accusations but in reality one who is suffering this kind of things are in villages who are not so much educated and but they do not make self this false accusations and they just bear the oppression. so this is the irony*
- *i , being a girl, can't believe how a girl, a woman who is daughter of one, mother for a child, a wife, a sister , moreover, a symbol of love and sympathy can do all these things just for their selfish motives...😧😧😧😧😧shame on all such women who ruin the lives of others,.....*

In the case of DistilRoBERTa for English sub-task B, it classified NGEN samples containing cuss words and gender words like *femnism*, *women*, *men* as GEN such as ones listed below. The third example below contains markdown, URLs and wrong spelling which might have also contributed to the misclassification. This suggests that further cleaning of the dataset is required.

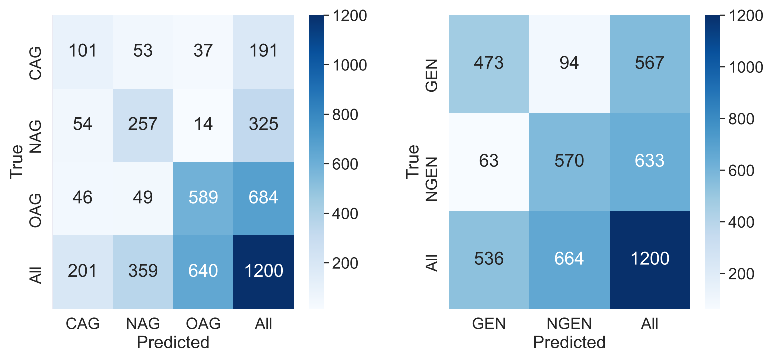
- *@queen bee says did i ask for ur sh*tty opinion??? f**king lil b**ch*

- *this is the truth about todays femnism .....womens are spiling life of so many mens today by fallsely accusing them. f**k this type of femnism..*
- *f**k femish*ts [#fkmkb] (http://www.youtube.com/results?search_query=%23fkmkb)*

5.3.2 Hindi

			Macro			
	Model	*Split*	*Precision*	*Recall*	*f1*	*Weighted-f1*
Sub-Task A	XLM-RoBERTa	Dev	0.6602	0.6376	0.6472	0.7146
		Test	0.7129	0.7269	0.7188	0.7927
Sub-Task B	XLM-RoBERTa	Dev	0.8125	0.7565	0.7801	0.8908
		Test	0.8704	0.8673	0.8683	0.8689

Table 5.4: Hindi Dev and Test Set Results



(a) Sub-Task A **(b)** Sub-Task B

Figure 5.5: Hindi Test Set Confusion Matrices

In Hindi sub-task A, the classifier misclassified the following covertly aggressive example:

- *awesome 😍but ..💔no one would give a "f" about this.. bas 2 din tak log dekhenge praise karenge or teesre din wohi chize.. ladkiyo ko hi faaltu ka support nilta rahega and ham hamesha "guilty" prove kiye jaate rahenge 💔.*
- *ha ha ha....* सारे मर्द एक जुट होकर धरना कर रहे है। महिलाओं से डर कर *!! 😊isse bolte hai kalyug!! [#kidding] (http://www.youtube.com/results?search_query=%23kidding)*

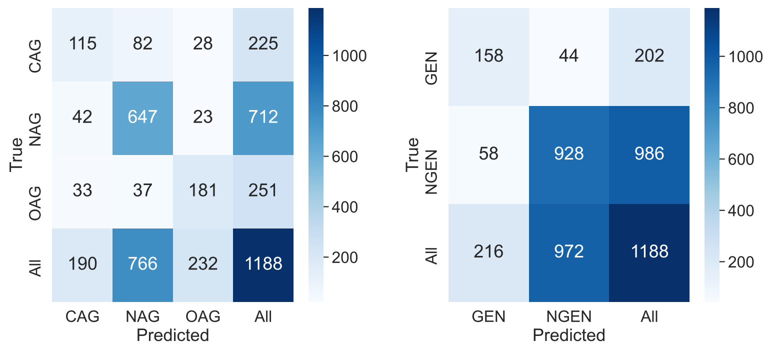
This could be because of presence of emojis such as 💔, 😍and veiled cuss words. The aggression is complex and required more context. It is also noteworthy that these samples are highly code-mixed. Similar misclassification can be seen for Hindi sub-task B. For example, the following are incorrectly predicted as gendered:

- *mu he ki ga**d . jadbhi bolti hi gu hi ugalti he.*
- *nikal pehli fursat me nikal*

5.3.3 Bengali

			Macro			
	Model	*Split*	*Precision*	*Recall*	*f1*	*Weighted-f1*
Sub-Task A	XLM-RoBERTa	Dev	0.7203	0.7121	0.7137	0.7539
		Test	0.7434	0.7136	0.7264	0.7880
Sub-Task B	XLM-RoBERTa	Dev	0.8580	0.8319	0.8439	0.9020
		Test	0.8431	0.8617	0.8519	0.9153

Table 5.5: Indian Bengali Dev and Test Set Results



(a) Sub-Task A **(b)** Sub-Task B

Figure 5.6: Bengali Test Set Confusion Matrices

In Bengali, sub-task A, covertly aggressive samples are again more difficult to classify. For instance in the following CAG comment there is an excess of emoji used which might have led it to be classified as NAG.

- *worst review* 👎👎

Interestingly there are code-mixed Hindi and complete English comments in the Bengali test set which were misclassified in sub-task B. For example:

- *madarc***d fake case walo ke*
- *mar sali k sathi rai k*
- *fat guy with salt and pepper hair plus plus (rajeev masand)half boiled egg*🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢🤢

5.4 Discussion and Conclusion

In this chapter, the thesis proposed automated methods for the detection of aggression in Hindi, English and Bengali languages. The thesis explored the usages of an attention head layer for making the classification instead of the typical fully connected layer. The performance of Checkpoint Ensembling is experimented with. In the English sub-task, the classifiers were unable to work on long text samples (§5.3.1). As future work, it will be interesting to try Longformer [75] to capture the long text samples. It can be speculated that further preprocessing and cleaning of the samples is also required as seen from the error analysis (§5.3.2).

Code-mixing is another interesting aspect of the dataset that has not been completely explored and exploited in this chapter, even though multi-lingual models were used. It was found that Hindi, Bengali and English languages were used interchangeably in all the three of the datasets. Measuring the amount of code-mixing and exploring models trained on a combination of datasets in different languages is worth exploring. Tackling the problem with a multitask model also seems promising.

The results suggest that Checkpoint Ensembling is a hit-or-miss approach when it comes to aggression identification. Further work is required in exploring its application and potential. Also, this chapter did not use many of the other ensemble techniques such as bagging, stacking, etc., which could be picked up for future work.

This work has been published as part of a larger study [76] in proceeding for TRAC-2 in LREC 2020. The code for all the work performed in this chapter is available github[1].

[1] https://github.com/cozek/trac2020_submission

Chapter 6

Using Text and Image Features to Classify Internet Memes

6.1 Introduction

Internet memes are typically images superimposed with text or text placed in a white space over an image [48]. They are one of the most widely shared content on social media. For instance, in Reddit, a site with over 330 million users [77], the r/memes[1] sub-reddit is the 2nd highest to gain the most subscribers [2] in 2019.

Although internet memes are often meant for humour or social commentary, it is also used as a means to propagate offensive, racist, sexist, and hateful content [78, 79]. Hence, to curb the spread of such unsavoury content, automated methods are required. This motivated the thesis's work in this chapter. The chapter uses the internet meme dataset called Memotion Analysis Dataset (released in SemEval 2020) [80] and proposes two DNN based classification systems for the multi-modal classification of internet memes.

The contributions of this chapter are as follows. Firstly, the thesis describes the classification tasks featured by the Memotion Analysis dataset and provides an analysis of the data samples in §6.2. Secondly, the thesis proposes two classification systems that use features extracted from pre-trained models. These pre-trained models and the classification systems built on top of them are described in §6.3 and §6.3.2 respectively. Thirdly, the performance of the classifiers are evaluated and their errors are analysed in §6.4. And finally, the chapter concludes with a discussion on the strengths and weaknesses of this study and future work in §6.5.

[1] https://www.reddit.com/r/memes

[2] https://subredditstats.com/r/memes

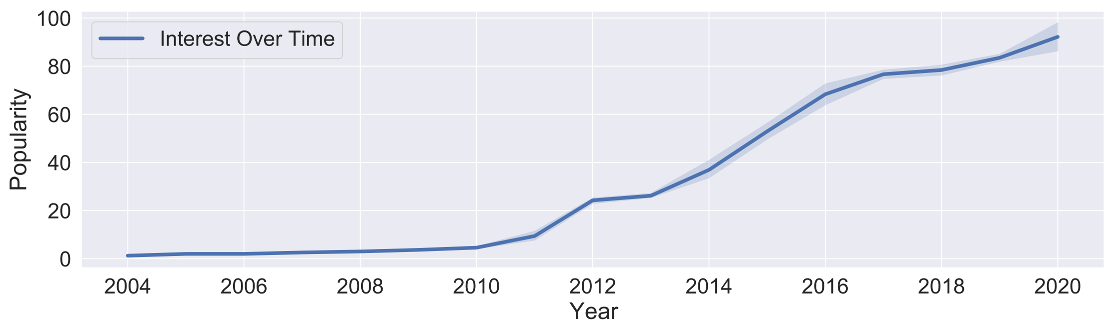


Figure 6.1: Search popularity of keyword ‘meme’ via Google Trends. A value of 100 indicates peak popularity

6.2 Memotion-2020 Dataset and Classification Tasks

The thesis proposes classification models for the first two of the three tasks ,i.e tasks A and B, for which the Memotion Analysis Dataset is annotated.

Classification Tasks

A) Sentiment Classification: Classify a meme as either positive, negative or neutral.

B) Humor Classification. Identify the type of humour expressed by a meme. The categories of humour include sarcastic, humorous, offensive and motivational. A meme might belong to one or more of these categories.

C) Scales of Semantic Classes. Quantify, in range 0 to 3, the extent to which a particular effect is being expressed. The effects being humour, sarcasm, offensive and motivational.

The metric of evaluation for Task A is *Macro-f1*. For Task B and C, the evaluation metric is the average of the macro-f1 for each of the humour categories. These performance metrics are used so as to be able to compare our models to the baseline systems proposed by the authors of the dataset. The dataset contained 6992 internet memes for training [80], i.e. images with their corresponding OCR extracted text (see Figure 6.2a). For evaluating our classification models, the dataset had a separate test set of 1878 memes. In Task A, the dataset is unbalanced for the negative class (see Table 6.1). For Task B, except for the motivational class, the positive samples outnumber the negative ones (see Table 6.2).

Preliminary analysis of the images in the dataset reveals that they are in various file formats with the majority of them being in *jpg* followed by the *png* format (see Table 6.3). On analysing further, it was observed that there is a large variance in the heights and widths of the images. This is also true for the lengths the text portion of the dataset (see Table 6.4). These findings suggest that there exists no particular standard for composition of memes. It also shows that data was sampled from various sources. These properties are perhaps magnified further when these memes hop through the internet from one platform to another due to users sharing them.

Label	*Positive*	*Negative*	*Neutral*
Count	4160	631	2201

Table 6.1: Task A Label Counts

Label		*Humor*	*Sarcasm*	*Offensive*	*Motivational*
Count	1	5341	5448	4279	2467
	0	1651	1544	2713	4525

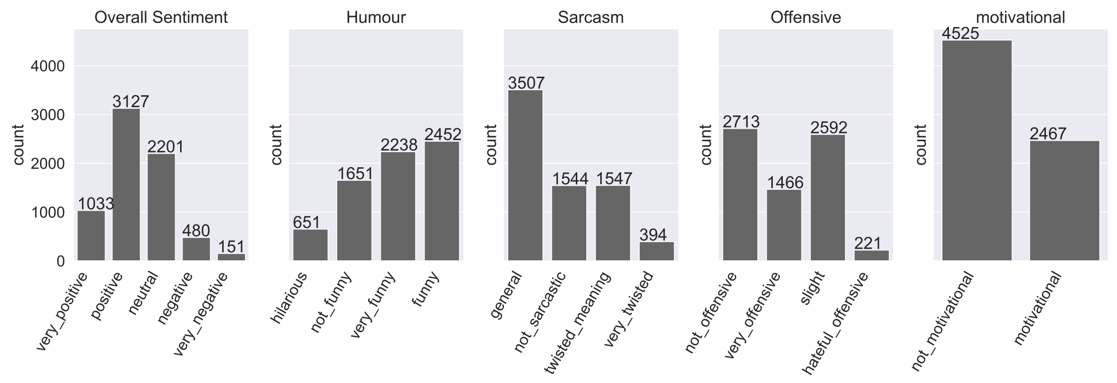
Table 6.2: Task B Label Counts. The Samples marked 0 are negative for their corresponding label

jpg	*jpeg*	*bmp*	*png*	*jpe*
4965	345	2	1679	1

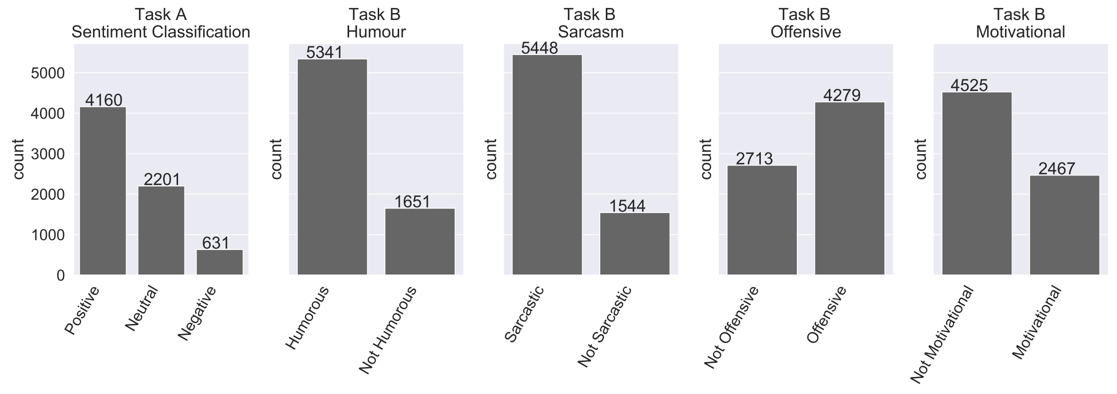
Table 6.3: Different image formats in the dataset

Feature	*Mean*	*Max*	*Min*	*Std*
Image Width (px)	587.06	4961	100	256.83
Image Height (px)	546.50	5553	123	250.04
Text Length (char)	82.78	1026	2	50.38
Text Length (words)	14.66	189	1	8.97

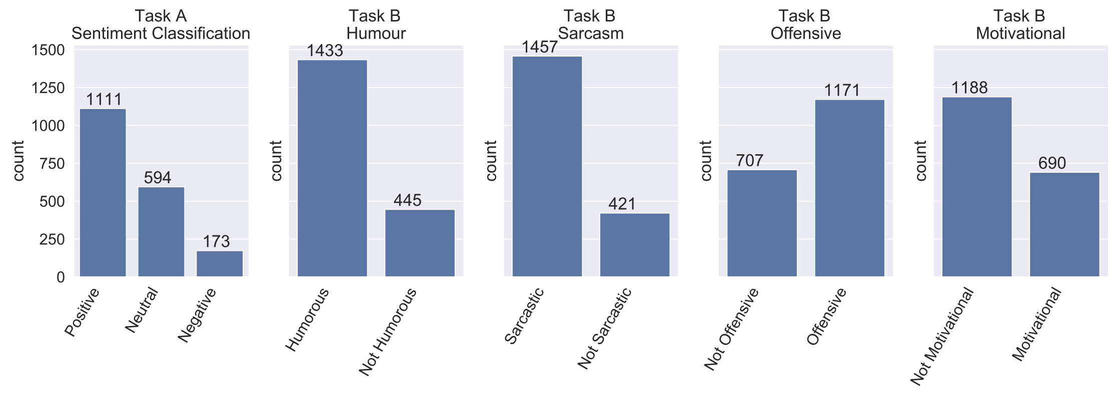
Table 6.4: Memotion Dataset Statistics



(a) Dataset Annotation



(b) Task A and B Train Set



(c) Task A and B Test Set

Figure 6.2: Memotion Dataset Countplots

6.3 Methodology

The chapter experiments with two DNN based classification systems for the classification tasks of the Memotion Dataset. The first system utilizes text and images features together while the second system uses only the image feature. Both of these models follow the typical neural network fine-tuning procedure. This procedure includes using a pre-trained model to extract features which are then fed into a fully connected layer that makes the classification. The classification systems are described in details in § 6.3.2.

For extracting features from the text and images, the thesis makes use of pre-trained DistilRoBERTa [62] and EfficientNets [81] respectively. These pre-trained models are described in §6.3.1. The main reason for selecting these models is their fast inference and training time while having high-performance benchmark scores in their domains.

6.3.1 Pre-trained Models

6.3.1.1 DistilRoBERTa

Distil* and RoBERTa have already discussed in §4.5.1 and §4.5.2. The classification systems in this chapter uses the pre-trained DistilRoBERTa implementation by Wolf et al. (2019) [66].

6.3.1.2 EfficientNets

EfficientNets [81] is a family of CNNs that have achieved state-of-the-art performance on the ImageNet challenge [82]. This family of CNNs are generally around 8× smaller and 6× faster on inference compared to other ImageNet models such as SENet [83], GPipe [84]. EfficientNets use a compound scaling method is to create the different models in the family that trade size for accuracy. The thesis tested the various models[3] in this family, from b0 to b7, and pick the best performing one for each classification task.

The primary motivation for using an ImageNet pre-trained network is because studies show such networks can be fine-tuned to attain performance improvements in other computer vision tasks [85].

6.3.2 Classification Systems

For training the models for each task, the chapter used 90% of the labelled data for training and rest 10% as validation. The splits are made preserving the percentage of samples for each class.

[3]https://github.com/lukemelas/EfficientNet-PyTorch

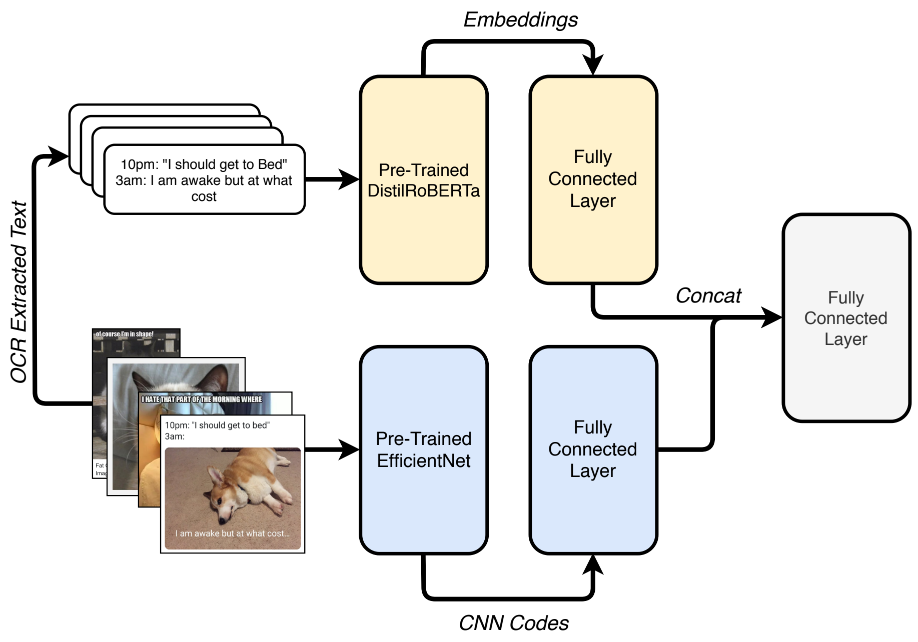


Figure 6.3: Classifier System Overview. System 1 uses both of the pre-trained model. System 2 only makes use of the images (colored in blue).

Both of the classification systems are built using the PyTorch [86] framework[4]. Both of these system were tested systems on Task A. But due to the sightly poorer performance of System 1 on Task A, System 1 is not used for Task B. Since Task B was multi-label, individual classifiers are trained for each of the classes, i.e, 1 classifier for each of Humor, Sarcasm, Offensive and Motivational. The details of the two classification systems are given in the subsubsections below.

6.3.2.1 System 1: DistilRoBERTa + EfficientNet

This system uses both EfficientNet and DistilRoBERTa[5] together for classifying the memes. Firstly, embeddings of the texts are extracted from pre-trained DistilRoBERTa. These embeddings are then passed through a fully connected layer L^1 with c-neurons, where c is the number of classes. Secondly, CNN codes are extracted from the final layer of pre-trained EfficientNet using the images, which are fed into another fully connected layer L^2, also having c neurons. Finally, the outputs of L^1 and L^2 are concatenated (column-wise) and fed to final fully connected layer L^F which makes the classification. A memory-efficient implementation of $swish$ [87] as the activation function for all of the aforementioned fully connected layers. Dropout [67] of 0.4 , 0.1, and 0.1 is used on L^1, L^2 and L^F respectively.

[4]https://pytorch.org/
[5]https://huggingface.co/transformers/model_doc/distilbert.html

6.3.2.2 System 2: EfficientNet

System 2 is largely similar to System 1, but only uses EfficientNet. The images are fed into the pre-trained EfficientNet Model to extract the CNN codes. These codes are then fed into a fully connected layer L^F having c neurons which makes the classification. A dropout of 0.2 and the same memory-efficient $swish$ activation function is applied on L^F. The chapter tested various scaled versions of EfficientNet for this model, b0 through b7. The best performing version were chosen for each of the individual classifiers built for Task B.

6.3.3 System Training and Inference

	Model		*Batch Size*	*Learning Rate*
Task A	DistilRoBERTa + EfficientNet_b4		72	$2e-4$
	EfficientNet_b4		300	$1e-4$
Task B	Humor	EfficientNet_b1	128	$1e-4$
	Sarcasm	EfficientNet_b0	300	$1e-4$
	Offensive	EfficientNet_b1	128	$1e-4$
	Motivational	EfficientNet_b4	128	$1e-4$

Table 6.5: Model Hyper-parameters

On both of the systems described above [6.3.2.1, 6.3.2.2], the entire model is fine-tuned, i.e all the layers including the pre-trained model, using a small learning rate coupled with a cross-entropy loss function. The learning rate and batch size for each model are given in Table 6.5. Both of the systems use $argmax$ on final layer outputs to make the predictions. System optimization is done using Ranger optimizer which is a combination of the RAdam [63] optimizer wrapped with LookAhead [64]. LookAhead's (k, α) are set to (5, 0.5). The systems are trained for 20 epochs on the training split with early-stopping. Early-stopping monitors the *Macro-f1* score of the system on the dev set and halts the training if this score fails to improve for 4 consecutive epochs. The weights of the network with the highest macro-f1 score on the dev split during training are stored and reloaded for making inference on the test set.

The performance of the best models found during training and experimentation are given in Table 6.6. The experiments show that using both the images and text together performs better than using images alone on the training set. But, the contrary is seen for the dev set albeit by a tiny margin. Both of these models underfit the dataset as is evident from their poor F1 scores. In Task B, even though only the images are used for classification, the performance is better than that of Task A. Hence, the image-only EfficientNet models are used for the final test set for both Task A and B.

			Macro-f1	
	Model		*Train Set*	*Dev Set*
Task A	DistilRoBERTa + EfficientNet_b4		0.6288	0.3536
	EfficientNet_b4		0.3429	0.3557
Task B	Humor	EfficientNet_b1	0.5004	0.5266
	Sarcasm	EfficientNet_b0	0.4915	0.5075
	Offensive	EfficientNet_b1	0.5087	0.5114
	Motivational	EfficientNet_b4	0.5010	0.5196

Table 6.6: Experiment Results

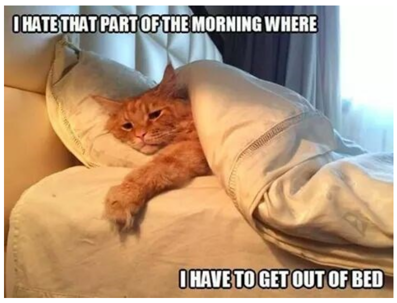
6.4 Result and Error Analysis

Since system 2 performed better than system 1 in the experiments, the chapter used only system 2, i.e. EfficientNet, on the test set. The official results on the test set are in Table 6.7. The confusion matrices for the same are given in Figure 6.6. In Task A, the EfficientNet scored a macro-f1 of 0.33, with the best model scoring 0.35. In Task B, it scored 0.5005 which is around 0.018 lower than the best and ranked 9th among all the systems evaluated so far by other researchers on this dataset (as recorded by the dataset authors [80]).

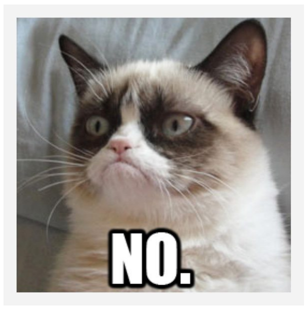
		Macro-f1			
	Model	*Author Baseline*	*Test Set*	*Best*	*Rank*
Task A	EfficientNet	0.2176	0.3286	0.3536	22
Task B	EfficientNet	0.5002	0.5005	0.5183	9

Table 6.7: Official Test Set Results

On analysing the classifications made by the System 1, it was found that system 1 is better at classifying the memes that had longer and more coherent sentences. But it fails to classify memes that had short and incoherent sentences. For example, System 1 correctly classified the meme in Figure 6.4a, but failed to classify the memes in Figure 6.4b and Figure 6.5a. This system also developed a strong tendency to classify the memes as positive in Task A (see Fig 6.6a). Perhaps due to more number of positive samples in the dataset. System 1 performed



(a) A cat in bed

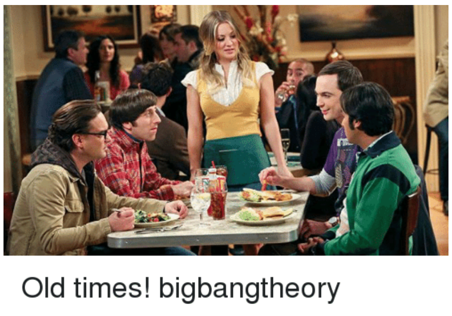


(b) Grumpy cat saying NO

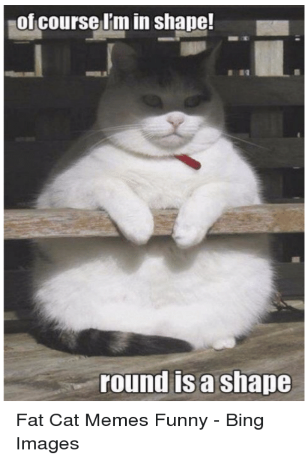
Figure 6.4: Meme Examples

poorly on memes that had multiple panels or when two or more memes were stitched together.

On the other hand, System 2 which uses only images is able to classify both Figure 6.4b and Figure 6.5a correctly. Further observation shows that in most of the samples, there is noise in the text such as URLs and unrelated sentences that might throw off the text-based classifier. Such URLs are often just for the promotion of the sites or users that host or create these memes. And being random, it is difficult to de-noise such text. But such samples are somewhat more correctly classified by System 2 which uses only the images. For example, the meme in Figure 6.5b is correctly classified by System 2 but System 1 fails. The *"Fat Cat Memes Funny - Bing Images"* part of the text includes references of the image source which is unrelated to the actual meme. This "noisy-ness" of the dataset perhaps might be the reason for System 2 having a higher macro f1 score. System 2 also failed to classify the samples where most of the context is in the text since it does not utilize any context from the text portion.

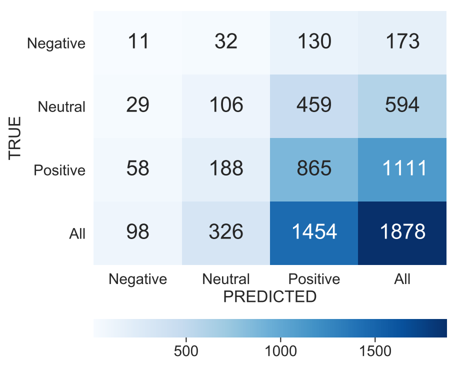


(a) Cast of Big Bang Theory

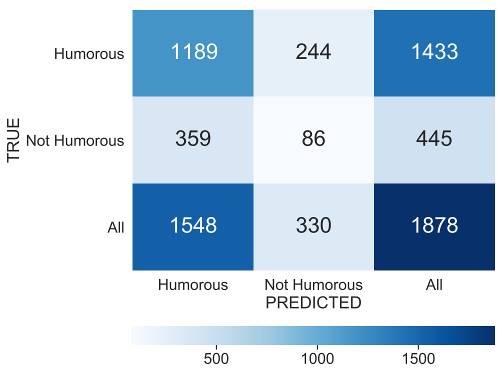


(b) Meme with noise in text

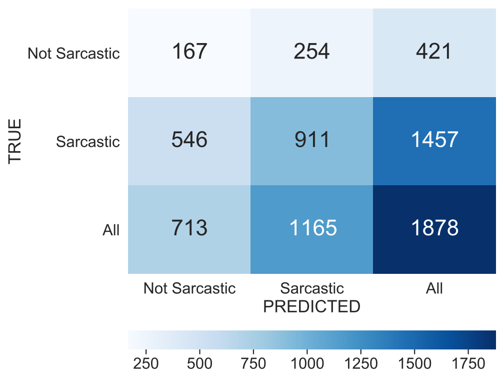
Figure 6.5: Misclassification Errors



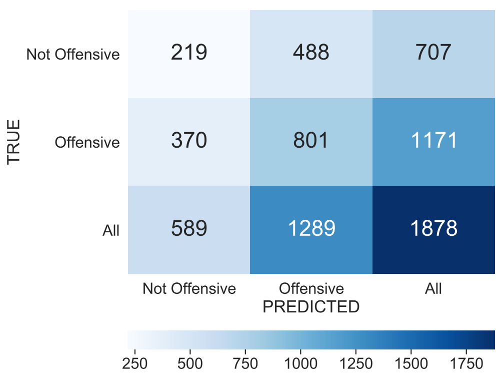
(a) Task A



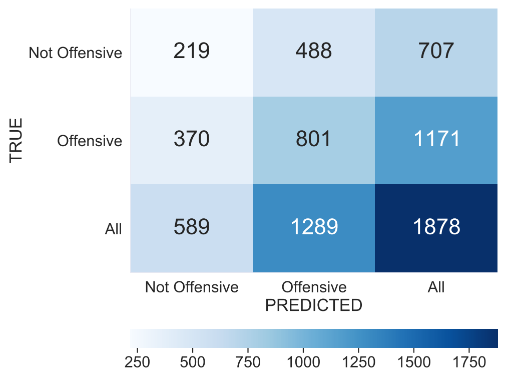
(b) Task B Humor



(c) Task B Sarcasm



(d) Task B Offensive



(e) Task B Motivational

Figure 6.6: Test Set Confusion Matrices

6.5 Discussion And Conclusion

This chapter built two types of systems for the task of classifying memes. One used both text and images together, while the other used only the images. Both of these systems relied on the features extracted from pre-trained networks. Results show that simply using the images performed better than using both image and text together. The strengths and weaknesses of both systems were discussed. It is obvious from the results that even the best model has underfitted the dataset, indicating that this field is currently at a nascent stage. It can be speculated that this will only get more difficult in the future as memes evolve towards using animations and videos. This is all the more reason to bring this field to the attention of the research community.

One weakness of this study is that it did not evaluate the performance of using only the text feature. The study also does not evaluate networks trained from scratch. Not to mention, none of prominent classical methods such as SVM, Logistic regression, etc were tested. All these must be picked up for future work, even if just for establishing a baseline. Furthermore, the classification tasks of Memotion require a multi-modal approach. As seen in the error analysis, misclassifications were made due to not having any context from the text and only relying on the images. This necessitates an investigation to answer why the multi-modal System 1 did not perform better than the uni-modal System 2 and how it can be improved.

The models and analysis code has been made available in github[6] reference of the kind reader.

[6]https://github.com/cozek/memotion2020-code

Chapter 7

Conclusion

This thesis looked at many different deep learning techniques for detecting hate speech in social media. Additionally, the thesis worked on the detection of online aggression and differentiating hate speech from other sentiments in internet memes. Chapter 2 listed many datasets and previous works in the the field of hate speech detection. In Chapter 3, the thesis built BiLSTM and CNN based models. These models used pre-trained embeddings to convert text into vector representations. We observe that these models did not perform well when linearly fine-tuned from one task to another. The simple minority oversampling technique, *interweaving*, did not work as expected.

Then from Chapter 3 onwards, the thesis moved on to transformer-based DNN models. Chapter 4 proposed three different models based on large transformers trained using the BERT technique and showed that they are fine candidates for transfer learning purposes. The thesis evaluated different pre-trained transformers such as RoBERTa, GPT-2, XLM-RoBERTa. When dealing with the large OffensEval dataset, the thesis used the fast and compressed Distil* transformers. The thesis also proposed the use of Checkpoint Ensembling which is useful for creating ensembles out of large models. These above mentioned methods were also applied for detection of online aggression in Chapter 5. For this task, the thesis proposed an Attention Head Layer as the classification layer. Finally, in Chapter 6, the thesis developed multi-modal and uni-modal systems for the detection of hate speech in internet memes using DistilRoBERTa and EfficientNet. was seen that the uni-modal image-only system performed better than the text-plus-image multi-modal system.

In total, the thesis worked on datasets in 4 different languages (English, Hindi, Indian Bengali and German) and 2 different domains, text and image. These datasets were recently released in different research workshops. The samples in these datasets were collected from various social media platforms which included Twitter, Facebook and YouTube. Notably, some of these datasets also contain a fair amount of code-mixing which required a multilingual approach. The thesis also released the code used for all the experiments in GitHub.

7.1 Future Work

In the thesis, only a single method of ensembling has been explored. As future work, of other ensembling techniques such as bagging and stacking could be evaluated. In this regard, the Meta Classifier Malmasi et al (2018) [68] might be a good starting point.

During experimentation, the huge training and inference time on OffensEval-2020's dataset proved to be quite a challenge. As such, it is important to explore techniques that could shorten the amount of time required to train such large models [88].

Also, in Chapter 6, it can be seen that the field of study dealing with detecting hate speech in internet memes has a lot of scope for improvement. The thesis aims to continue its efforts on building and experimenting with multi-modal models for the same. For this, the Hateful Memes Challenge by Facebook[1] seems to be a promising interesting next stop.

[1] https://ai.facebook.com/hatefulmemes

Bibliography

[1] Hugh Chen, Scott Lundberg, and Su-In Lee. Checkpoint ensembles: Ensemble methods from a single training process, 2017.

[2] Wikipedia. Social media. Available at https://en.wikipedia.org/wiki/Social_media. Accessed: 2019-11-19.

[3] Paul Mozur. A genocide incited on facebook, with posts from myanmar's military. Available at https://www.nytimes.com/2018/10/15/technology/myanmar-facebook-genocide.html. Accessed: 2019-11-19.

[4] Hanna Kozlowska. Sri lanka shut down facebook, whatsapp, and instagram to stop anti-muslim violence. Available at https://qz.com/1223787/sri-lanka-shut-down-facebook-whatsapp-and-instagram-to-stop-anti-muslim-violence/. Accessed: 2019-11-19.

[5] BBC. Germany starts enforcing hate speech law. Available at https://www.bbc.com/news/technology-42510868. Accessed: 2019-11-19.

[6] Aurelien Breeden. France will debate a bill to stop online hate speech. what's at stake? Available at https://www.nytimes.com/2019/07/01/world/europe/france-bill-to-stop-online-hate-speech.html.

[7] Facebook AI Research Team. AI advances to better detect hate speech. https://ai.facebook.com/blog/ai-advances-to-better-detect-hate-speech, 2020. Accessed: 2020-05-28.

[8] John T. Nockleby et. al. *Encyclopedia of the American Constitution*. Macmillan, New York NY, 2nd edition, 2000.

[9] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *ICWSM*, 2017.

[10] Chikashi Nobata, Joel R. Tetreault, Achint Oommen Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *WWW*, 2016.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *WWW*, 2015.

[15] Zeerak Waseem and Dirk 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, June 2016. Association for Computational Linguistics.

[16] Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. Aggression-annotated Corpus of Hindi-English Code-mixed Data. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA).

[17] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *SemEval@NAACL-HLT*, 2019.

[18] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *ArXiv*, abs/1405.4053, 2014.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[20] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[21] Shervin Malmasi and Marcos Zampieri. Challenges in discriminating profanity from hate speech. *J. Exp. Theor. Artif. Intell.*, 30:187–202, 2017.

[22] Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, 2010.

[23] Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[24] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada, August 2017. Association for Computational Linguistics.

[25] Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. All you need is ”love”: Evading hate-speech detection. *ArXiv*, abs/1808.09115, 2018.

[26] Segun Taofeek Aroyehun and Alexander Gelbukh. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[28] Ignacio Arroyo-Fernández, Dominic Forest, Juan-Manuel Torres-Moreno, Mauricio Carrasco-Ruiz, Thomas Legeleux, and Karen Joannette. Cyberbullying detection task: the ebsi-lia-unam system (elu) at coling’18 trac-1. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 140–149, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[29] Kashyap Raiyani, Teresa Gonçalves, Paulo Quaresma, and Vitor Beires Nogueira. Fully connected neural network with advance preprocessor to identify aggression over Facebook and twitter. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 28–41, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[30] Marian-Andrei Rizoiu, Tianyu Wang, Gabriela Ferraro, and Hanna Suominen. Transfer learning for hate speech detection in social media, 2019.

[31] Arup Baruah, Ferdous A. Barbhuiya, and Kuntal Dey. Abaruah at semeval-2019 task 5 : Bi-directional lstm for hate speech detection. In *SemEval@NAACL-HLT*, 2019.

[32] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[33] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[34] Vijayasaradhi Indurthi, Bakhtiyar Syed, Manish Shrivastava, Nikhil Chakravartula, Manish Gupta, and Vijay K. Varma. Fermi at semeval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in twitter. In *SemEval@NAACL-HLT*, 2019.

[35] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[36] Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *ArXiv*, abs/1803.01400, 2018.

[37] Alexandre Salle and Aline Villavicencio. Incorporating subword information into matrix factorization word embeddings. *ArXiv*, abs/1805.03710, 2018.

[38] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *ArXiv*, abs/1803.11175, 2018.

[39] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. Deep contextualized word representations. *ArXiv*, abs/1802.05365, 2018.

[40] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

[41] Shubhanshu Mishra and Sudhanshu Mishra. 3Idiots at HASOC 2019: Fine-tuning Transformer Neural Networks for Hate Speech Identification in Indo-European Languages. In *Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation*, FIRE 2019 Working Notes, pages 208–213, Kolkata, India, December 2019. CUER Workshop Proceedings.

[42] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya 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.

[43] Sudhanshu Mishra, Shivangi Prasad, and Shubhanshu Mishra. Multilingual joint fine-tuning of transformer models for identifying trolling, aggression and cyberbullying at TRAC 2020. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 120–125, Marseille, France, May 2020. European Language Resources Association (ELRA).

[44] Benet Oriol Sabat, Cristian Canton-Ferrer, and Xavier Giró-i-Nieto. Hate speech in pixels: Detection of offensive memes towards automatic moderation. *CoRR*, abs/1910.02334, 2019.

[45] Julian Risch and Ralf Krestel. Bagging BERT models for robust aggression identification. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 55–61, Marseille, France, May 2020. European Language Resources Association (ELRA).

[46] Shiladitya Bhattacharya, Siddharth Singh, Ritesh Kumar, Akanksha Bansal, Akash Bhagat, Yogesh Dawer, Bornini Lahiri, and Atul Kr. Ojha. Developing a multilingual annotated corpus of misogyny and aggression. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 158–168, Marseille, France, May 2020. European Language Resources Association (ELRA).

[47] Akshi Kumar, Kathiravan Srinivasan, Wen-Huang Cheng, and Albert Y. Zomaya. Hybrid context enriched deep learning model for fine-grained sentiment analysis in textual and visual semiotic modality social data. *Inf. Process. Manag.*, 57(1), 2020.

[48] David M Beskow, Sumeet Kumar, and Kathleen M Carley. The evolution of political memes: Detecting and characterizing internet memes with multi-modal deep learning. *Information Processing & Management*, 57(2):102170, 2020.

[49] Thomas Mandl, Sandip Modha, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. Overview of the HASOC track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In *Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation*, December 2019.

[50] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2000.

[51] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[52] François Chollet et al. Keras. https://keras.io, 2015.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[54] Sandip Modha, Thomas Mandl, Prasenjit Majumder, and Daksh Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *FIRE ’19*, 2019.

[55] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2019.

[56] Kaushik Amar Das and Ferdous Ahmed Barbhuiya. Team FalsePostive at HASOC 2019: Transfer-Learning for Detection and Classification of Hate Speech. In *Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation*, FIRE 2019 Working Notes, pages 320–327, Kolkata, India, December 2019. CEUR-WS.org.

[57] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*, 2020.

[58] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[59] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *In the Proceedings of ICLR.*, 2019.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison,

Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[61] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[63] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[64] Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back, 2019.

[65] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[67] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[68] Shervin Malmasi and Marcos Zampieri. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:187 – 202, 2018.

[69] Jeff Schultz. How much data is created on the internet each day? Available at https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/. Accessed: 2020-05-19.

[70] L. Gao, A. Kuppersmith, and R. Huang. Recognizing Explicit and Implicit Hate Speech Using a Weakly Supervised Two-path Bootstrapping Approach. In *IJCNLP 2017*, pages 774–782, Taipei, Taiwan, 2017.

[71] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany, August 2016. Association for Computational Linguistics.

[72] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? comparing deep learning activation functions across nlp tasks, 2019.

[73] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[74] Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. Evaluating aggression identification in social media. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 1–5, Marseille, France, May 2020. European Language Resources Association (ELRA).

[75] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[76] Arup Baruah, Kaushik Amar Das, Ferdous Ahmed Barbhuiya, and Kuntal Dey. Aggression identification in english, hindi and bangla text using bert, roberta and svm. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 76–82, Marseille, France, May 2020. European Language Resources Association (ELRA).

[77] Arielle Pardes. The Inside Story of Reddit's Redesign. https://www.wired.com/story/reddit-redesign/, 2018. Accessed: 2020-04-18.

[78] J Williams. The good and the bad of political memes during election seasons. *The Chronicle, Independent News Organization at Duke's University*, 2016.

[79] Jessica Drakett, Bridgette Rickett, Katy Day, and Kate Milnes. Old jokes, new media – online sexism and constructions of gender in internet memes. *Feminism & Psychology*, 28(1):109–127, 2018.

[80] Chhavi Sharma, William Paka, Scott, Deepesh Bhageria, Amitava Das, Soujanya Poria, Tanmoy Chakraborty, and Björn Gambäck. Task Report: Memotion Analysis 1.0 @SemEval 2020: The Visuo-Lingual Metaphor! In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, Sep 2020. Association for Computational Linguistics.

[81] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019.

[82] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[83] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2017.

[84] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2018.

[85] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better?, 2018.

[86] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[87] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7, 2017.

[88] Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: Adaptive sparsity by fine-tuning, 2020.

Author's Biography

Kaushik Amar Das received his B. Tech degree in Computer Science and Engineering from Royal School of Engineering and Technology in 2018. He has been pursuing M. Tech at the Department of Computer Science and Engineering, IIIT Guwahati, since July 2018.

Publications made out of this thesis
(listed in reverse chronological order)

1. Kaushik Amar Das, Arup Baruah, Ferdous Ahmed Barbhuiya and Kuntal Dey. KAFK at SemEval-2020 Task 8: Extracting Features From Pre-trained Neural Networks To Classify Internet Memes. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, COLING '20, Barcelona, Spain, December, 2020. Association for Computational Linguistics. **Submitted for publication, currently under review**

2. Kaushik Amar Das, Arup Baruah, Ferdous Ahmed Barbhuiya and Kuntal Dey. KAFK at SemEval-2020 Task 12: Checkpoint Ensemble of Transformers for Hate Speech Classification". In *Proceedings of the 14th International Workshop on Semantic Evaluation*, COLING '20, Barcelona, Spain, December, 2020. Association for Computational Linguistics. **Submitted for publication, currently under review**

3. Arup Baruah, Kaushik Amar Das, Ferdous Ahmed Barbhuiya and Kuntal Dey. Aggression Identification in English, Hindi and Bangla Text using BERT, RoBERTa and SVM. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, LREC '20, pages 76–82, Marseille, France, May, 2020. European Language Resources Association. online https://www.aclweb.org/anthology/2020.trac-1.12.pdf

4. Kaushik Amar Das, Ferdous Ahmed Barbhuiya. Team FalsePostive at HASOC 2019: Transfer-Learning for Detection and Classification of Hate Speech. In *Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation*, FIRE'19, Kolkata, India, December, 2019. CEUR-WS.org, online http://ceur-ws.org/Vol-2517/T3-19.pdf