Hate speech detection against women and immigrants

Akshat Chaudhary [1]*, **Maitra Patel** [1] , **Aditya Arya** [1]
[1]Indian Institute of Technology Delhi

Abstract

In this project, we attempt the English part of the SemEval 2019 Task 5 about the detection of hate speech against women and immigrants in English messages extracted from Twitter. The task is organized in two related classification subtasks: a main binary subtask for detecting the presence of hate speech, and a finer-grained one devoted to identifying further features in hateful contents such as the aggressive attitude and the target harassed, to distinguish if the incitement is against an individual rather than a group.

1 Introduction

Hate Speech (HS) is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics (Nockleby et al., 2000). Given the huge amount of user-generated contents on the Web, and in particular on social media, the problem of detecting, and therefore possibly contrasting the HS diffusion, is becoming fundamental, for instance for fighting against misogyny and xenophobia.

2 Data

In this competition we have been provided with a training and testing data separately with 5999 [1] and 2000 samples respectively. The training data is annotated and contains the columns `id`, `text`, `HS` (Hate Speech), `TR` (Targeted) and `AG` (Aggressive). Each of these label columns are binary. The distribution of the tweets for the training data is shown as in Figure 2. As we can see only less than half of the samples labelled as `HS` fall under `TR` or `AG`. Hence, we require a good discriminative model for the same. Since, each sample can be associated with multiple labels, this is a multi-label classification problem. In Figure 1 we also present the distribution of labels among the training samples. It is clear from the figure that more than half the training data consists of non-hateful speech, so a sampling strategy can be employed such that we have approximately equal number of labels across all samples.

2.1 Text Preprocessing

Given the source of the texts from Twitter, the dataset contains a substantial number of hashtags, user mentions, URLs, and emoticons. To ensure the consistency and uniformity of the text data, a preprocessing pipeline is employed. Firstly, URLs are replaced with the token `<url>` to abstract their content. Similarly, hashtags are replaced with `<hashtag>` to preserve the contextual information while removing the specific hashtag identifier. User mentions are transformed into the generic token `<user>` followed by '@' symbol to maintain the structure of the mentions. Emoticons, being an integral part of informal text communication, are replaced with the token `<emoticon>` to standardize their representation. Additionally, numeric terms and their variations (e.g., ordinal indicators like 'st', 'nd', 'rd', 'th') are substituted with `<number>` to generalize numerical information. Finally, the text is tokenized using a tokenizer that preserves case sensitivity and reduces repeated characters to maintain readability and consistency across the dataset. An example of processed text looks like:
<user> you're the exception you weren't a rude lil cunt like some hoes <emoticon> <emoticon>

3 Methodology

3.1 Logistic Regression

In this section, we detail the application of logistic regression for hate speech detection, targeting the classification of hate speech (HS), targeted speech (TR), and aggressive language (AG). The process

*{mt6210814,ph1210813,mt6210958}@iitd.ac.in
[1]1 row was corrupted

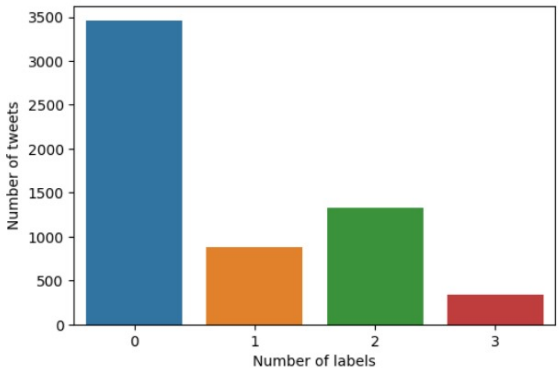

Figure 1: Distribution of labels in the training data.

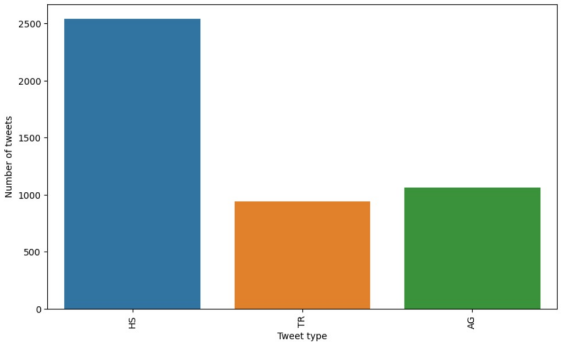

Figure 2: Distribution of tweets in the training data.

encompasses model training, evaluation, and prediction.

3.1.1 Model Training

The logistic regression model is employed as the primary classification algorithm for hate speech detection. The training process involves the following steps:

1. **Feature Engineering**: Utilizing the TF-IDF vectorizer, the textual data is transformed into numerical feature vectors. The TF-IDF vectorization scheme captures the importance of terms within the corpus.

2. **Data Splitting**: The dataset is split into training and validation sets using a stratified approach, maintaining the class distribution ratio. The split ratio adopted is 80:20, ensuring sufficient data for model training and evaluation.

3. **Model Initialization**: An instance of logistic regression is initialized with default parameters for each target variable (HS, TR, AG).

4. **Model Training**: The logistic regression model is trained on the training data, with the feature vectors as input and the corresponding binary labels (0 or 1) for hate speech as the target variable.

3.1.2 Model Evaluation

Following model training, the performance of the logistic regression classifier is evaluated using standard classification metrics. The evaluation process includes the following steps:

1. **Prediction Generation**: The trained logistic regression model is utilized to generate predictions for the validation set. Each tweet in the validation set is classified as either containing hate speech or not.

2. **Metric Calculation**: The accuracy of the model is computed as the proportion of correctly classified instances over the total number of instances in the validation set. Additionally, precision, recall, and F1-score are calculated to assess the model's performance across different classes.

3.1.3 Model Prediction

Upon successful evaluation, the trained logistic regression model is applied to predict the labels

for hate speech, targeted speech, and aggressive language in the test dataset. The prediction process involves the following steps:

1. **Preprocessing**: The same preprocessing steps applied to the training data are replicated on the test data to ensure consistency in feature extraction.

2. **Feature Extraction**: The TF-IDF vectorizer fitted on the training data is used to transform the test text into feature vectors.

3. **Prediction Generation**: The logistic regression model generates predictions for hate speech, targeted speech, and aggressive language labels in the test dataset.

4. **Submission File Creation**: The predicted labels are organized into a submission file in CSV format, containing the tweet IDs along with the corresponding predictions for each target variable.

3.2 BERT

In this subsection, we present the implementation of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) for hate speech detection. BERT is a pre-trained language representation model that has achieved state-of-the-art performance on various natural language processing tasks. We fine-tune the pre-trained BERT model for our specific hate speech classification task.

3.2.1 Model Architecture

The BERT model architecture consists of 12 transformer layers for the base model and 24 transformer layers for the large model. Each transformer layer comprises self-attention mechanisms and feed-forward neural networks. Specifically, we utilize the BERT-base uncased model, which consists of 12 transformer layers with a hidden size of 768 and 12 self-attention heads.

To adapt BERT for hate speech classification, we append a classification layer on top of the BERT model. The classification layer consists of a dropout layer, followed by two fully connected layers with ReLU activation functions. Finally, a softmax layer is applied to compute the probabilities of the input belonging to each class. The architecture is summarized as follows:

```
BERTM_Arch(
  (bert): BertModel(
    (embeddings): BertEmbeddings(...)
    (encoder): BertEncoder(...)
    (pooler): BertPooler(...)
  )
 (dropout): Dropout(p=0.1, inplace=False)
  (relu): ReLU()
  (fc1): Linear(in_features=768,
                out_features=512,
                bias=True)
  (fc2): Linear(in_features=512,
                out_features=n_class,
                bias=True)
  (softmax): LogSoftmax(dim=1)
)
```

where:

- `BertModel`: Pre-trained BERT model consisting of embeddings, transformer encoder, and pooler.
- `Dropout`: Dropout layer to prevent overfitting.
- `ReLU`: Rectified Linear Unit activation function.
- `Linear`: Fully connected layers for classification.
- `LogSoftmax`: Logarithmic softmax function for probability computation.

The input to the BERT model consists of tokenized text sequences, including special tokens [CLS] and [SEP], along with attention masks to indicate the presence of valid tokens. During forward propagation, the BERT model processes the input tokens through multiple transformer layers, and the final hidden state corresponding to the [CLS] token is extracted for classification. The extracted representation is passed through the classification layer to predict the probabilities of hate speech classes.

3.3 Transformer-based Text-to-Text Transfer Transformer (T5)

In this subsection we use the T5 model (Raffel et al., 2020) introduced by Google, which is an encoder-decoder model, and pre-trained on large amount of data.

3.3.1 Configuration

The T5 model is configured using a predefined set of parameters to ensure consistency and reproducibility across experiments. These parameters are given in Table 1

Table 1: T5 Model Configuration

Parameter	Value
Seed	42
Model Path	t5-base [2]
Tokenizer	T5 Tokenizer
Source Maximum Length	250
Target Maximum Length	20
Batch Size	16
Validation Split	0.25
Device	GPU P100
Full Finetuning	True
Learning Rate	3e-5
Optimizer	AdamW
Criterion	BCEWithLogitsLoss
Save Best Only	True
Number of Validation Checks	3 during training
Epochs	1

3.3.2 T5 Dataset

The `T5 Dataset` class is responsible for processing the dataset and preparing it for training, validation, and testing. It performs the following tasks:

- Extracts texts and labels from the dataset based on provided indices.
- Tokenizes the input texts using the T5 tokenizer.
- Ensures that input texts are padded or truncated to the specified maximum length.
- Generates input tensors for source texts and target labels.
- Provides methods to retrieve source input IDs, attention masks, target input IDs, and target attention masks.

3.3.3 T5 Model Architecture

The `T5 Model` class defines the architecture of the T5 model used for text-to-text transfer learning. It consists of:

- A pre-trained T5 model loaded from the specified model path.
- Forward method implementation for performing inference and training.
- Inputs include source input IDs, attention masks, decoder input IDs, decoder attention masks, and labels.
- Outputs include model predictions and computed loss, if labels are provided.

The T5 model architecture follows the standard transformer-based architecture, comprising encoder and decoder layers. The specific details of these layers are encapsulated within the pre-trained T5 model, which is loaded from the specified model path. Implementation inspired from (Jaunjale, 2022)

3.4 LightGBM

In this subsection, we employ the LightGBM (Light Gradient Boosting Machine) algorithm (Ke et al., 2017) for multilabel classification tasks. LightGBM is a powerful gradient boosting framework that is efficient in handling large-scale datasets and offers high accuracy with faster training speed compared to other algorithms.

We follow a systematic approach outlined below:

1. **Data Preprocessing**: We start by preprocessing the text data and labels. The text data is transformed into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. This converts the textual information into a format suitable for machine learning algorithms. Additionally, we split the dataset into training and validation sets, with an 80-20% ratio, to facilitate model training and evaluation.
2. **Model Training with Cross-Validation**: Next, we employ k-fold cross-validation to train and evaluate the LightGBM model. This technique partitions the dataset into k equally sized folds, with each fold used once as a validation set while the remaining $k-1$ folds form the training set. We iterate through each fold, training a LightGBM model on the training data and evaluating its performance on the validation data. This process allows us to assess the robustness and generalization of the model across different subsets of the dataset.

[2] Pre-trained T5-base model available at https://huggingface.co/google-t5/t5-base

3. **Evaluation Metrics**: During each fold of cross-validation, we compute two key evaluation metrics: accuracy and Hamming loss. Accuracy measures the proportion of correctly predicted labels, providing insight into the overall predictive performance of the model. Hamming loss quantifies the average fraction of labels that are incorrectly predicted, offering a more nuanced assessment of the model's performance, especially in multilabel classification scenarios.

4. **Average Performance Metrics**: Finally, we aggregate the evaluation metrics obtained from each fold of cross-validation to compute the average accuracy and average Hamming loss across all folds. These average metrics serve as indicators of the overall performance and effectiveness of the LightGBM model in handling the multilabel classification task.

By following this approach, we aim to leverage the capabilities of LightGBM to effectively classify text data into multiple categories, thereby addressing the complex nature of multi-label classification problems like in our case.

4 Results & Discussion

In this section, we discuss the results of the various models as discussed in Section 3.
The performance metrics of the vanilla logistic regression model on the 20% validation set is provided in Table 2. All the metrics have been calculated for the 0/1 label separately, and the results are not so bad considering it is vanilla logistic regression. We get an average f1-score of 0.75 for HS, 0.74 for TR, and 0.65 for AG class.

Table 2: Performance Metrics of Logistic Regression

Metric	HS	TR	AG
Accuracy	0.7708	0.885	0.860
Precision (0)	0.78	0.90	0.87
Precision (1)	0.75	0.75	0.77
Recall (0)	0.85	0.97	0.98
Recall (1)	0.66	0.44	0.25
F1-Score (0)	0.81	0.93	0.92
F1-Score (1)	0.70	0.55	0.38

The performance for the BERT model is given in Table 3. We can see precision, recall, and F1-score of 0 for the AG label, which might be caused due to the data imbalance after splitting of train dataset into validation. Overall, the performance of the model is not expected, it can be improved by applying a CNN or LSTM instead of a FNN, or adjusting the hyperparameters appropriately.

Table 3: Performance Metrics of BERT Model

Metric	HS	AG	TR
Accuracy	0.73	0.80	0.87
Precision (0)	0.74	0.80	0.88
Precision (1)	0.70	0.00	0.79
Recall (0)	0.86	1.00	0.99
Recall (1)	0.50	0.00	0.24
F1-Score (0)	0.80	0.89	0.93
F1-Score (1)	0.59	0.00	0.37

As for the case of T5 transformer, we could not get the results accurately because of class imbalance, and some errors which could not be resolved before the deadline.
For the LGBM model, we got an average accuracy of 63% on the validation set. This was the model finally selected for hidden test cases on Kaggle, with a score of 0.5.
Overall, in the leaderboard (preliminary results), our team was 5th out of 7 teams. We could not run more experiments, due to time and resource constraints, but given the time, did whatever was possible.
All the code can be accessed at the following: https://github.com/ac031203/ELL884-Project

5 Scope for Improvement

We could directly use BERT models fine-tuned on large amount of hate-speech detection subtask like HateBERT (Caselli et al., 2021), and instead of using Neural Networks, club them with Bidirectional LSTM as proposed in (Pan et al., 2021) which has shown to perform with f1-score as high as 93%.
Several ensemble techniques, like soft voting, maximum value, hard voting and stacking, can be applied to increase the accuracy as presented in (Mnassri et al., 2022).
The currently publicly available datasets for Hate Speech Detection are **Davidson** (Davidson et al., 2017), **HatEval** (Basile et al., 2019), and **OLID** (Zampieri et al., 2019). A dataset combining all these three datasets, will be much more balanced, and thus help us in creating much accurate models.
Furthermore, advanced graphical models like Graph Convolutional Networks (GCN) can also

be used as surveyed in (Pandey, 2023)

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, pages 54–63.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Prithvi Jaunjale. 2022. T5 multi-label text classification. Accessed: 06/05/24.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, and Qiwei Ye. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

Khouloud Mnassri, Praboda Rajapaksha, Reza Farahbakhsh, and Noel Crespi. 2022. Bert-based ensemble approaches for hate speech detection. *arXiv preprint arXiv:2209.06505*.

John T Nockleby, Leonard W Levy, Kenneth L Karst, and Dennis J Mahoney. 2000. Encyclopedia of the american constitution. *Hate Speech; Levy, L., Karst, K., Eds*, pages 1277–1279.

Feiyu Pan, Xingwen Liu, Junwen Liang, Zixuan Yu, Shiliang Pu, Zhenyu Zhang, and Wenge Rong. 2021. A transformer-based method for multi-label text classification.

Shasank Sekhar Pandey. 2023. A comparative study of bert-cnn and gcn for hate speech detection. *University of Twente*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Mohan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*.