UNIVERSITÀ DEGLI STUDI DI MILANO

Hate Speech Detection

By

Delnavaz Fotouhi

Prof. Alfio Ferrara

Text Mining

Department of Computer Science

Wednesday 15th January, 2025

Contents

List of Figures

Abstract

Hate speech detection is critical for applications like controversial event extraction, building AI chatterbots, content recommendation, and sentiment analysis. The goal of the project is to create models for classifying text contents as offensive. This study focuses on a multilabel classification problem, addressing five categories of hate speech: *NotHate*, *Racist*, *Sexist*, *Homophobe*, and *OtherHate*.

Chapter 1

Introduction

The rapid growth of social media and digital communication platforms has brought opportunities for global interaction. However, it has also led to the spread of hate speech. Hate speech includes offensive language targeting individuals or groups based on attributes such as race, gender, or sexual orientation. Detecting hate speech is a critical step toward creating safer digital spaces.

This report focuses on the development of machine learning models for hate speech detection as a multilabel classification task. The goal is to predict five categories of hate speech—*NotHate*, *Racist*, *Sexist*, *Homophobe*, and *OtherHate*—using a dataset of tweets.

Three models were employed in this study: Naive Bayes, Logistic Regression, and BERT. Naive Bayes is a probabilistic classifier often used for text classification due to its simplicity and speed. Logistic Regression, a linear model, offers a scalable and interpretable approach to classification tasks. BERT, a transformer-based model, leverages contextual embeddings to capture semantic relationships in text, providing state-of-the-art performance in natural language processing tasks.

This report provides a comprehensive evaluation of these models, highlighting their strengths and limitations. The results contribute to the broader effort of advancing hate speech detection systems, enabling safer and more inclusive online interactions.

Chapter 2

Dataset

The dataset used for this project is the **MMHS150K Dataset**, a multimodal hate speech dataset containing 150,000 tweets. Each tweet includes both text and an associated image. For this project, only the textual content of the tweets was utilized.[1]

2.1 Dataset Description

The MMHS150K dataset was constructed by gathering tweets using the Twitter API over six months, from September 2018 to February 2019. These tweets were annotated via Amazon Mechanical Turk (AMT). Annotators were provided with definitions of hate speech and examples to ensure consistency. Each tweet was classified into one of six categories: NotHate, Racist, Sexist, Homophobic, Religion, and OtherHate.[2]

Each tweet was labeled by three annotators to reduce discrepancies. The dataset includes the following fields:

- `tweet_url`: The URL of the tweet.
- `labels`: An array of three numeric labels (0–5) indicating the classification from each annotator.
- `img_url`: The URL of the associated image.
- `tweet_text`: The text content of the tweet.

- `labels_str`: An array of string labels corresponding to `labels`.

The images and tweet URLs included in the original dataset were eliminated to focus exclusively on the text-based classification task. The dataset was loaded and processed by extracting the `tweet_text`, `labels`, and `labels_str` fields into a structured format for analysis.

2.2 Exploratory Analysis

To gain insights into the dataset, the following analyses were conducted:

Dataset Size and Composition

The dataset comprises 149,823 entries with three columns: `tweet_text`, `labels`, and `labels_str`.

The `labels_str` column contains 232 unique combinations, with the most common being `[NotHate, NotHate, NotHate]`, representing 57,890 tweets.

2.3 Label Distribution

The `labels_str` column was exploded to create individual entries for each label. The breakdown of labels is as follows:

Label	**Count**	**Percentage (%)**
NotHate	312,039	69.4
Racist	63,543	14.1
OtherHate	31,548	7.0
Sexist	22,805	5.1
Homophobe	16,932	3.8
Religion	2,607	0.6

Table 2.1: Distribution of labels in dataset

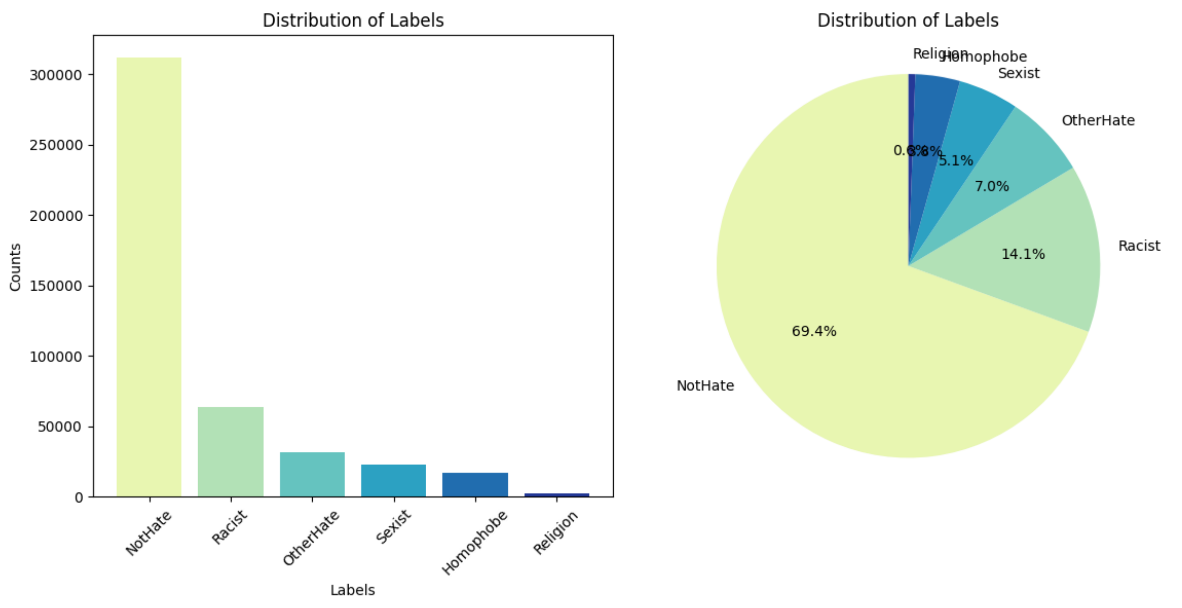


Figure 2.1: Labels Distribution

These distributions highlight the predominance of non-hate speech tweets in the dataset, followed by racist and other hate speech categories.

2.4 Tweet Length Analysis

The length of each tweet was calculated, and its distribution was analyzed across different labels. A histogram demonstrated variations in tweet length by category.

This comprehensive exploration of the dataset laid the foundation for subsequent data manipulation and model development phases.

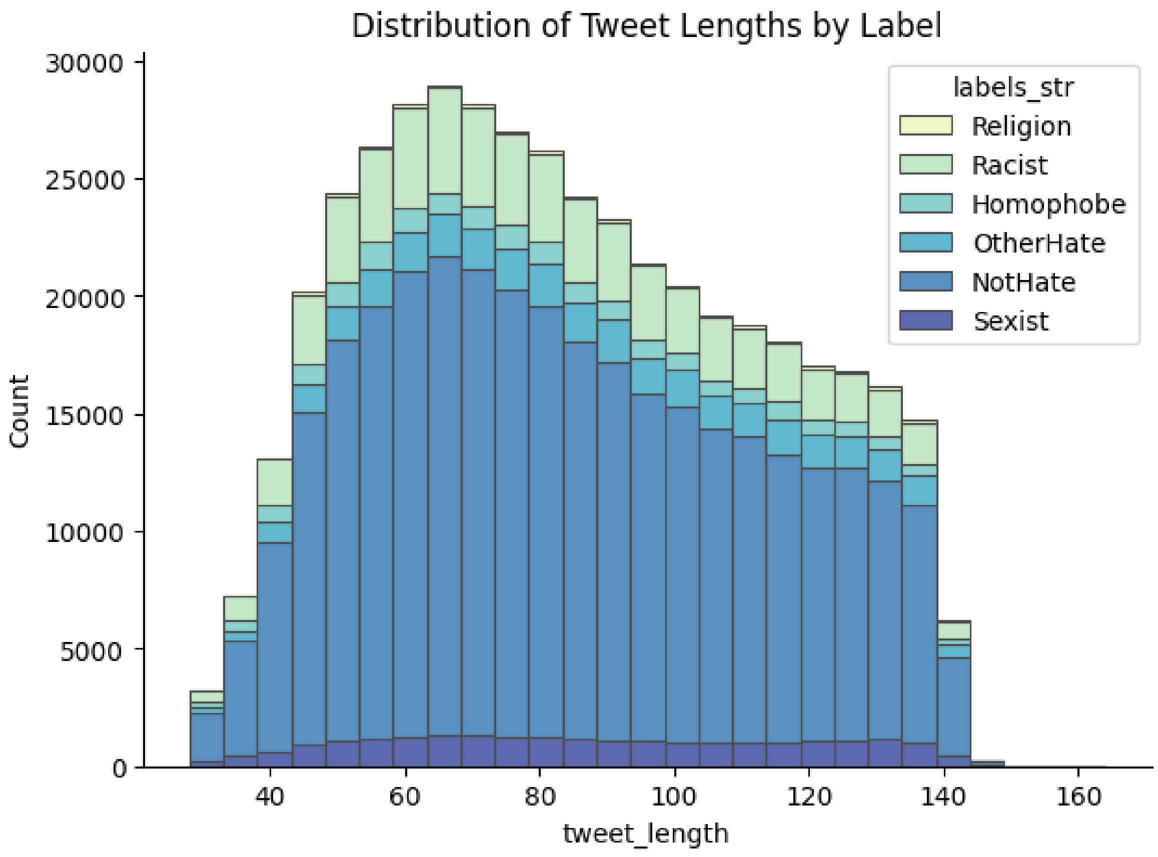


Figure 2.2: Tweet Length Distribution

Chapter 3

Data Manipulation

In this section, we discuss the steps taken to manipulate the data to make it suitable for the classification task, including label modification, encoding, balancing, and filtering.

3.1 Data Imbalance Handling

The original dataset contained a small number of tweets labeled as ”Religion,” making it a highly imbalanced class. To address this issue, the ”Religion” label was merged with the ”OtherHate” category. Specifically, all occurrences of the ”Religion” label were replaced with ”OtherHate.” This modification helped reduce the imbalance in the dataset, making the classification problem more manageable. The labels for each tweet were updated accordingly.

In addition to that, the dataset originally exhibited a significant imbalance, with the majority of tweets labeled as ”NotHate.” To eliviate the effect of this imbalance, tweets that were labeled exclusively as ”NotHate” were removed from the dataset. This filtering step ensured that only tweets containing hate speech remained for model training, improving the performance of the classifier by focusing on the more relevant data.

3.2 Label Encoding

Since this task involves multilabel classification, where each tweet can belong to multiple categories simultaneously, the labels were encoded into a binary format. In multilabel classification, the goal is to predict multiple independent binary labels for each sample. Therefore, each label was transformed into a binary vector, where each position in the vector corresponds to a particular class, and the value is 1 if the label is present and 0 if it is absent. This method allowed us to treat the classification task as a series of independent binary classification problems, one for each label. The encoding process resulted in a new column, `encoded_labels`, where each entry contains a binary list representing the presence or absence of each label in the tweet.

This encoding approach is essential for multilabel classification tasks, as it enables the model to independently predict each label and handle cases where multiple labels are relevant to a single instance.

3.3 Tweet Length Filtering

The next step involved filtering tweets based on their length. Since most tweets in the dataset had fewer than 140 characters, tweets longer than 140 characters were considered outliers and removed. This allowed for more uniform data.

3.4 Final Dataset Overview

After the data manipulation steps, the final dataset was prepared for analysis and model development. The distribution of labels in the filtered dataset was analyzed again to ensure the changes had been effective. The label counts were visualized, which provided a clear overview of the labels after the data manipulation steps.

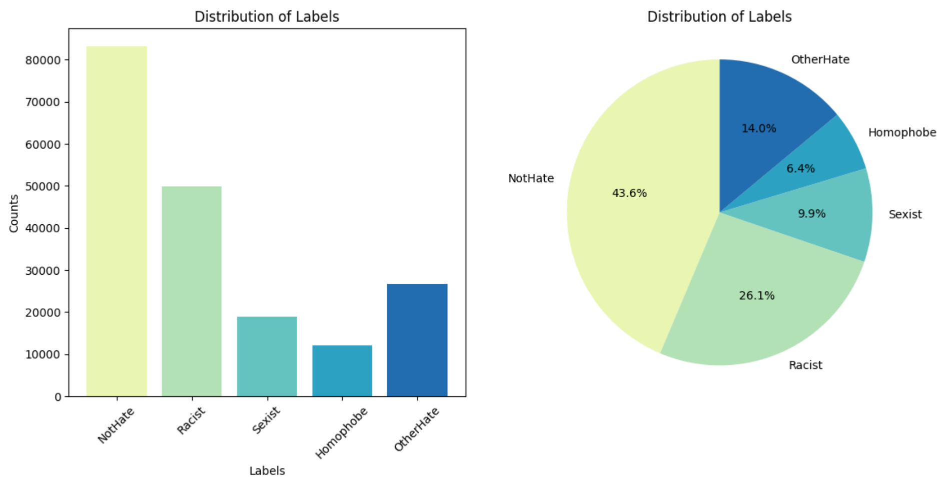


Figure 3.1: Labels Distribution after Data Manipulation

3.5 Label Co-occurrence Analysis

In addition to label distribution, the co-occurrence of labels was analyzed using a cross-tabulation matrix. This matrix showed how often pairs of labels appeared together in the same tweet. This analysis revealed interesting patterns, such as the frequent co-occurrence of "Racist" and "NotHate" labels.

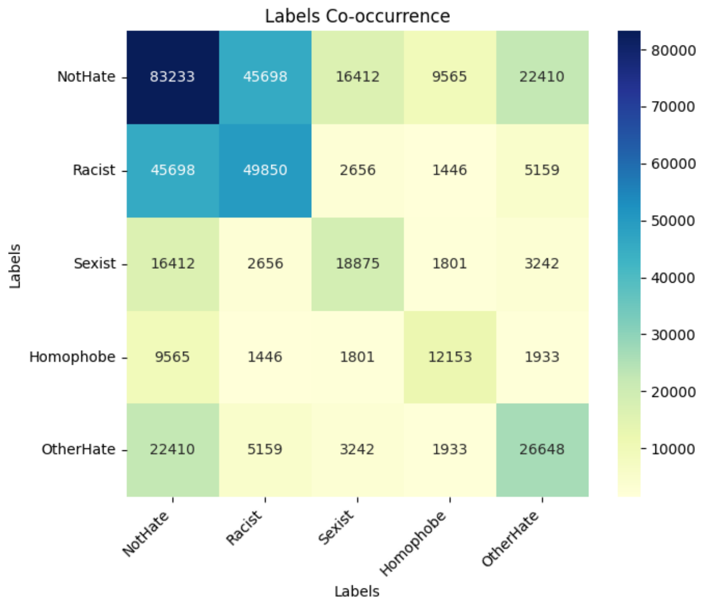


Figure 3.2: Label Co-occurrence

Chapter 4

Data preparation

Before feeding the data into machine learning models, several preprocessing steps were applied to the dataset to enhance the quality of the input and optimize model performance.

4.1 Text Cleaning

The text data in the dataset was cleaned to remove unnecessary elements and improve the quality of the features used for training. This process involved several steps:

- **Lowercasing**: All text was converted to lowercase to ensure uniformity and prevent the model from treating the same word with different cases as different words.
- **Removal of Mentions and URLs**: Mentions (i.e., "@username") and URLs were removed from the text as they are not useful for detecting hate speech.
- **Removal of Numbers and Punctuation**: Numbers and punctuation marks were removed, as they generally do not contribute to the classification task.
- **Stopword Removal**: Common stopwords, which do not carry significant meaning (e.g., "the", "is", "in"), were removed to reduce noise in the text.

- **Non-ASCII Characters Removal**: Non-ASCII characters, including emojis, were eliminated to prevent issues during vectorization.
- **Lemmatization**: Words were lemmatized to their root form using the WordNet lemmatizer. This process converts different inflections of a word (e.g., ”running”, ”ran”, ”runs”) into a common base form (e.g., ”run”), which helps the model generalize better.

These steps were implemented using the Python Natural Language Toolkit (NLTK) library, and the cleaned text was stored in a new column, `clean`, in the DataFrame.

4.2 Text Vectorization

After cleaning the text data, it was transformed into numerical representations using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. The TF-IDF method is based on two key components:

- **Term Frequency (TF)**: This is the frequency of a word in a document. It is calculated as:

  $$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

  A higher term frequency indicates that the term is more relevant in that document.
- **Inverse Document Frequency (IDF)**: This measures the importance of the word across all documents in the corpus. It helps to reduce the weight of common words and give more weight to rare but informative words. It is calculated as:

  $$\mathrm{IDF}(t) = \log\left(\frac{N}{\mathrm{df}(t)}\right)$$

  where N is the total number of documents, and $\mathrm{df}(t)$ is the number of documents that contain the term t. A higher IDF score indicates that the word is rarer across documents.

The TF-IDF score for each term is the product of TF and IDF:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

This score reflects both the importance of a term in a specific document and its relative rarity across the entire corpus.

4.2.1 Feature Selection

In the vectorization process, the text data was converted into a matrix where each row represents a document (in this case, a tweet) and each column corresponds to a unique word or n-gram (a sequence of words). The values in the matrix are the TF-IDF scores for each word in the document. Since the vocabulary of the dataset can be large, a subset of the most important features (words) was selected to optimize the model's performance. The following feature selection parameters were applied:

- **Max Features**: Only the top 1000 most informative features (words and n-grams) were selected based on their TF-IDF scores. This reduces the dimensionality of the dataset and focuses the model on the most relevant terms.
- **N-grams**: Both unigrams (single words) and bigrams (pairs of consecutive words) were considered as features. This helps capture more contextual information and word dependencies that can be important for detecting hate speech. For example, a bigram like "hate speech" could provide more insight into the sentiment of the tweet than individual words like "hate" or "speech" on their own.
- **Minimum Document Frequency**: Words that appeared in fewer than 5 documents were excluded from the feature set. This removes very rare words that are unlikely to provide valuable information for the classification task.

The resulting feature set consisted of 1000 features, representing the most significant unigrams and bigrams in the dataset. These features were then used to create the TF-IDF matrix, which was subsequently used as the input for the machine learning models.

4.3 Splitting the Dataset

The dataset was then split into training, validation, and test sets to evaluate the model's performance. The data was divided as follows:

- 80% of the data was allocated for training.
- 10% was used for validation, which is essential for tuning hyperparameters and avoiding overfitting.
- The remaining 10% was reserved for testing, to evaluate the model's generalization performance on unseen data.

The data was randomly shuffled before splitting to ensure a representative distribution of labels in each subset.

Chapter 5

Model Training and Evaluation

5.1 Naive Bayes

Algorithm Overview

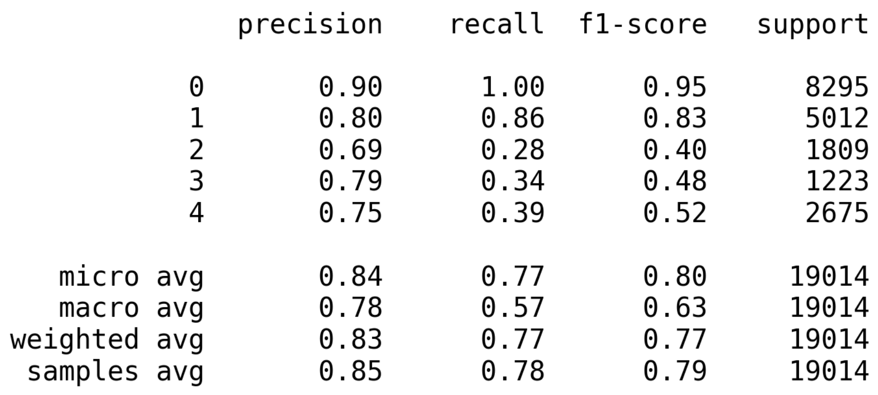
The Naive Bayes algorithm is a probabilistic classifier based on Bayes' theorem, assuming that the features are conditionally independent given the class. Despite its simplicity, Naive Bayes has proven effective for text classification tasks, especially when the data has high dimensionality.

For this study, the Multinomial Naive Bayes variant was used, which is particularly suited for discrete data such as word frequencies or TF-IDF scores. The algorithm calculates the posterior probability of a document belonging to a specific class based on the observed features and assigns it to the class with the highest probability.

The key advantages of Naive Bayes include its computational efficiency, scalability to large datasets, and robust performance on text-based tasks. However, its assumption of feature independence might limit its effectiveness when features have strong dependencies.

Results and Discussion

The performance of the Naive Bayes classifier was evaluated using the classification report, per-label accuracies, and subset accuracy. Below are the results obtained:



```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      8295
           1       0.80      0.86      0.83      5012
           2       0.69      0.28      0.40      1809
           3       0.79      0.34      0.48      1223
           4       0.75      0.39      0.52      2675

   micro avg       0.84      0.77      0.80     19014
   macro avg       0.78      0.57      0.63     19014
weighted avg       0.83      0.77      0.77     19014
 samples avg       0.85      0.78      0.79     19014
```

Figure 5.1: Naive Bayes Classification Report

- **Precision, Recall, and F1-Score:** The classifier achieved high precision and recall for the "NotHate" class, indicating its ability to accurately identify non-hate speech tweets. However, the performance on minority classes, such as "Sexist" and "Homophobe", was lower.

- **Per-Label Accuracies:**

Label	Accuracy (%)
NotHate	90.36%
Racist	80.26%
OtherHate	78.42%
Sexist	83.37%
Homophobe	89.99%

Table 5.1: Per-Label Accuracy

These results highlight the effectiveness of the model in predicting more frequent labels but reveal the challenges in accurately predicting less frequent ones.

- **Subset Accuracy:** The subset accuracy, which measures the exact match ratio of the predicted labels to the true labels, was 50.52%. This metric is stricter than per-label metrics and reflects the difficulty of the multilabel classification task, especially when the number of co-occurring labels is high.

In summary, the Naive Bayes classifier offered a simple yet effective approach for hate speech detection. Despite its limitations, it served as a valuable benchmark for comparing more advanced models in subsequent sections.

5.2 Logistic Regression Classifier

Algorithm Overview

Logistic Regression is a linear model commonly used for classification tasks. It predicts the probability that a given input belongs to a particular class by modeling the relationship between the input features and the class labels using a logistic (sigmoid) function.

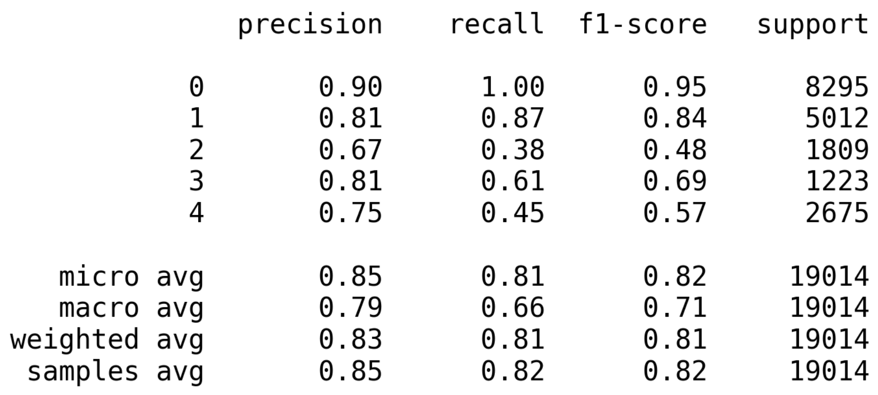
For this multilabel classification task, the One-vs-Rest (OvR) strategy was employed. OvR trains one binary classifier for each label, treating each label as a separate classification problem while ignoring others. This approach allows Logistic Regression to handle multiple labels simultaneously, making it well-suited for multilabel classification problems.

Logistic Regression is effective for its simplicity, interpretability, and scalability to large datasets. It performs well when the data is linearly separable but may struggle with non-linear decision boundaries.

Results and Discussion

The Logistic Regression model was evaluated using standard metrics, including precision, recall, F1-score, per-label accuracies, and subset accuracy. Below are the results:

- **Precision, Recall, and F1-Score:** The classifier achieved a high F1-score for the "NotHate" label and good performance for the "Racist" label. However, the F1-scores for minority labels such as "Sexist" and "OtherHate" were comparatively lower, indicating challenges in handling minority classes.



```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      8295
           1       0.81      0.87      0.84      5012
           2       0.67      0.38      0.48      1809
           3       0.81      0.61      0.69      1223
           4       0.75      0.45      0.57      2675

   micro avg       0.85      0.81      0.82     19014
   macro avg       0.79      0.66      0.71     19014
weighted avg       0.83      0.81      0.81     19014
 samples avg       0.85      0.82      0.82     19014
```

Figure 5.2: Logistic Regression Classification Report

- **Per-Label Accuracies:** The accuracies for individual labels are summarized in Table 5.2.

Label	**Accuracy (%)**
NotHate	90.34
Racist	82.21
Sexist	84.07
Homophobe	92.82
OtherHate	79.76

Table 5.2: Per-Label Accuracies for Logistic Regression Classifier

  These results indicate strong performance on frequent labels and relatively good performance on less frequent ones compared to Naive Bayes.

- **Subset Accuracy:** The subset accuracy was 55.56%, reflecting the proportion of samples where all predicted labels matched the true labels exactly. This metric demonstrates a moderate improvement compared to Naive Bayes.

- **Micro and Macro Averages:**
  - Micro Avg $F1$-Score: 82%

  - Macro Avg $F1$-Score: 71%

  The micro-average places greater weight on the performance of frequent labels, while the macro-average treats all labels equally, highlighting the model's challenge in handling imbalanced datasets.

In summary, the Logistic Regression model outperformed Naive Bayes in terms of subset accuracy and per-label metrics, particularly for minority classes.

5.3 BERT-Based Model

To enhance the performance of hate speech detection, a transformer-based model, BERT (Bidirectional Encoder Representations from Transformers), was employed.

BERT utilizes a transformer architecture, which enables it to capture contextual information from both the left and right contexts of a word in a sentence. This bidirectional context encoding is particularly beneficial for tasks, where understanding the surrounding context is crucial for accurate classification. In our implementation, we used the bert-base-uncased variant of BERT, which consists of 12 transformer layers and 768 hidden units.

Training Process

The BERT model was fine-tuned on the dataset for a multi-label classification task. Tweets were tokenized using the BERT tokenizer, ensuring uniform length through padding and truncation. The encoded tokens were fed into the pre-trained BERT model (`bert-base-uncased`) with an added classification head to handle five output labels. The model was trained for 5 epochs with the AdamW optimizer, mixed-precision training using GradScaler, and binary cross-entropy loss with logits. The training and validation datasets were created with an 80-10-10 split, and data were fed into the model in batches of size 64.

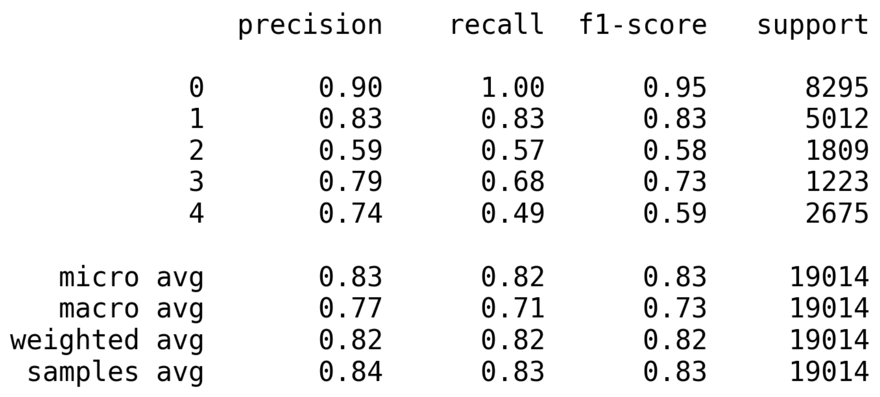
Results and Discussion

The BERT model achieved a subset accuracy of 56.51%, outperforming the previous models (Naive Bayes and Logistic Regression). Per-label accuracies were calculated as follows:

Label	**Accuracy (%)**
NotHate	90.37
Racist	81.17
Sexist	83.62
Homophobe	93.37
OtherHate	80.00

Table 5.3: Per-Label Accuracies for BERT Model

Additionally, the classification report showed high precision, recall, and F1 scores for the dominant label (*NotHate*) and moderate performance for minority labels such as *OtherHate*. The macro-average F1 score was 73%, indicating that the model effectively balanced performance across all labels.



```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      8295
           1       0.83      0.83      0.83      5012
           2       0.59      0.57      0.58      1809
           3       0.79      0.68      0.73      1223
           4       0.74      0.49      0.59      2675

   micro avg       0.83      0.82      0.83     19014
   macro avg       0.77      0.71      0.73     19014
weighted avg       0.82      0.82      0.82     19014
 samples avg       0.84      0.83      0.83     19014
```

Figure 5.3: Bert Classification Report

Chapter 6

Conclusion

In this study, we explored three models: Naive Bayes, Logistic Regression, and BERT—for the task of hate speech detection as a multilabel classification problem. Each model was evaluated based on its performance in predicting five different hate speech categories: *NotHate*, *Racist*, *Sexist*, *Homophobe*, and *OtherHate*.

The results demonstrated that BERT and Logistic Regression outperformed Naive Bayes in terms of per-label accuracy, as illustrated in Figure 6.1. BERT achieved the highest overall accuracy across most labels, showcasing its ability to capture semantic relationships in the data due to its contextual embeddings. However, its computational complexity, higher training time, and resource demands make it less practical for scenarios where efficiency and cost are key constraints.

On the other hand, Logistic Regression provided competitive results with a simpler and more resource-efficient approach. It proved to be a viable alternative to BERT, particularly in scenarios requiring faster deployment and lower computational overhead.

Naive Bayes, while the least accurate, offers simplicity and speed, making it suitable for preliminary analyses or resource-constrained applications. However, its limited capability to capture complex linguistic patterns restricted its effectiveness in detecting certain hate speech categories.

In conclusion, the choice of model depends on the specific application context. For projects prioritizing accuracy and capable of accommodating higher computational costs, BERT is the preferred choice. Conversely, for use cases requiring cost-efficiency and speed, Logistic Regression is a strong contender that balances simplicity with performance.

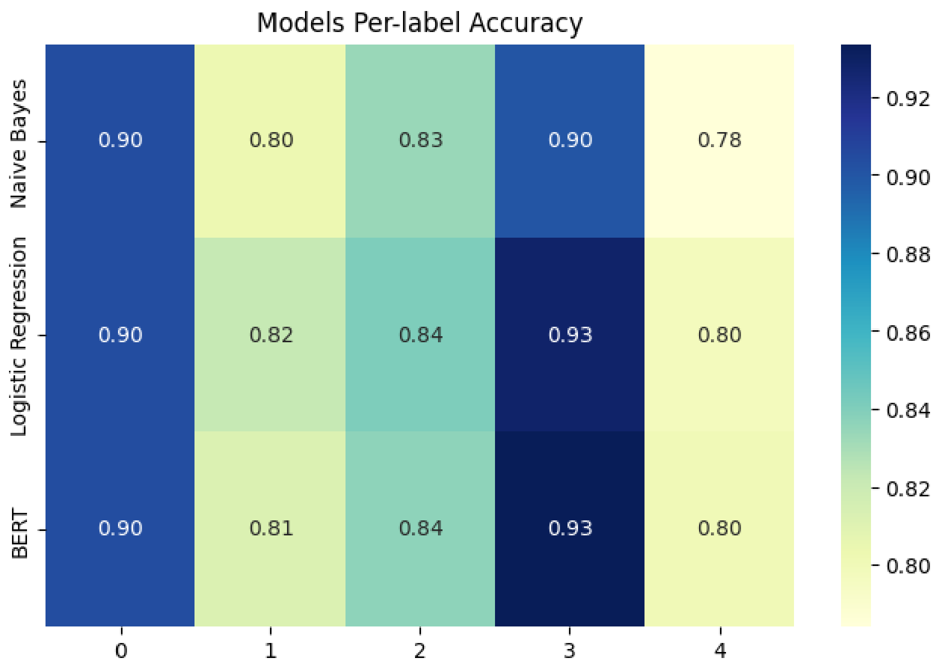


Figure 6.1: Per-label Accuracy for All Models

Bibliography

[1] Multimodal hate speech dataset. https://www.kaggle.com/datasets/victorcallejasf/multimodal-hate-speech/data?select=MMHS150K_GT.json.

[2] Lluis Gomez Dimosthenis Karatzas2 Raul Gomez1, Jaume Gibert1. Exploring hate speech detection in multimodal publications. page 8, 2019.