Human Language Technologies Project Report

Anile Alessia (619554)
Russo Giuseppe Gabriele (583744)
Ambrosi Dylan Nico (578586)
Piccolo Chiara (619224)

Master Degree Computer Science
Human Language Technologies 2023-2024

Introduction

Bullying, described as unwanted aggressive behavior causing physical, mental, or social harm, has a virtual counterpart known as cyberbullying or online bullying, manifesting through texts or images. The increase in cyberbullying cases, facilitated by social media platforms like Facebook, Instagram, and Twitter, highlights the need for robust detection systems.

Our project aims to explore the efficacy of three distinct machine learning models—Naive Bayes, LSTM, and Transformers—in detecting cyberbullying within tweets. We seek to determine not only whether a tweet contains bullying but also to classify the type of bullying present. This comparative analysis will highlight the strengths and weaknesses of each model in dealing with the nuanced domain of cyberbullying. By evaluating the performance of these models, we aim to provide insights into their applicability in real-world scenarios, contributing to the development of more effective tools for cyberbullying detection and classification.

Previous Works

This section reviews notable contributions in the field of machine learning-based cyberbullying detection and highlights the distinctiveness of our current work through a comparative analysis.

Previous research has explored various machine learning models for cyberbullying detection with notable success. Naïve Bayes classifiers, known for their simplicity and effectiveness, have demonstrated considerable accuracy in identifying bullying behaviors in online interactions. For instance, Naïve Bayes models have achieved high accuracy in detecting cyberbullying content on platforms like Twitter, indicating their utility in this domain [5].

Long Short-Term Memory (LSTM) networks, including both unidirectional and bidirectional variants, have shown superior performance in capturing temporal dependencies and context in text sequences, which is crucial for detecting nuanced patterns of abusive language over time. Research has highlighted that bidirectional LSTMs (BiLSTMs) enhance context understanding by processing the input text in both forward and backward directions, leading to improved detection accuracy [9].

Transformer-based models like BERT have revolutionized natural language processing by using bidirectional context and self-attention mechanisms to capture intricate semantic relationships within the text. Studies have shown that BERT significantly improves the precision and recall of cyberbullying detection systems due to its deep contextual understanding [17].

Similarly, other transformer models like RoBERTa and ELECTRA [7] have been employed for cyberbullying detection. These models refine the pre-training objectives of BERT, leading to better performance and efficiency in various text classification tasks, including detecting cyberbullying [1].

Task

This project addresses a text classification problem within the field of Natural Language Processing. The objective is to develop models that can effectively categorize tweets based on their content, specifically in relation to bullying. The classification tasks are defined as follows.

The first task involves classifying tweets into various categories that represent different types of bullying. Each category is distinct and requires the model to identify specific elements within the text that correspond to types of bullying behaviors. This task is complex due to the need for the classifier to accurately distinguish between multiple, nuanced categories that reflect the diverse nature of bullying.

The second task simplifies the classification process by reducing the categories to two: 'bullying' and 'non-bullying'. This binary classification approach is aimed at determining the presence or absence of bullying behavior in tweets.

Data

Dataset Analysis

Our initial dataset, available on Kaggle [10], contained 47,000 tweets evenly distributed across several categories of bullying, with approximately 8,000 tweets in each category: 'age', 'ethnicity', 'religion', 'gender', 'other cyberbullying', and 'non-cyberbullying' (Appendix, 1). Upon analyzing duplicates in the dataset, we noticed that most tweets categorized as 'non-cyberbullying' were also categorized as 'other cyberbullying'. Given the generic nature and ambiguity of the 'other cyberbullying' category, we decided to remove it to enhance the clarity and specificity of our data categorization.

Several analyses were conducted on the dataset, starting with the creation of word clouds for each category of cyberbullying, to identify the most relevant words for each (Appendix, Figure 3). This proved to be highly beneficial as it allowed us to pinpoint the frequent appearance of the abbreviation 'mkr' within the 'non-cyberbullying' category tweets. Importantly, this abbreviation does not signify a relevant expression; rather, it refers to the Australian cooking show 'My Kitchen Rules.' This discovery was crucial as it could have potentially influenced the analysis outcomes.

Additional analyses were carried out on the tweet dataset, focusing on the quantity and length of the words used. Tweets are generally brief, which aligns with Twitter's intended use as a platform for quick and succinct communication. The average length per tweet was 31 words per tweet, showing that users tend to express ideas or opinions concisely. Furthermore, three-quarters of the tweets contained 44 words or fewer, ensuring that even the more verbose tweets remained relatively concise (Appendix, Figure 3). The average word length in tweets was less than 4 characters, signifying a frequent use of short words—a common practice in spaces where brevity is essential. A quarter of the words in tweets were only 2 characters long, reflecting the prevalent use of abbreviations and acronyms (Appendix, Figure 3).

These statistics highlighted some of the major challenges in applying cyberbullying detection to tweets. The brevity and informal nature of tweets, characterized by the use of abbreviations and slang, poses significant challenges in accurately understanding and processing these texts. Furthermore, the limited length of tweets restricts the available context, making it more difficult to accurately interpret the content. These factors complicate the development of effective cyberbullying detection algorithms [2].

Dataset split and preprocessing

The dataset was split into an 80% training set and a 20% test set, with 20% of the training set designated as a validation set. The preprocessing approach varied based on the model due to their distinct capabilities in handling text data.

General preprocessing steps were applied across all models, which included cleaning tweets by removing elements like links, mentions, and hashtags, converting all text to lowercase, and stripping out special characters, extra spaces, textual emoticons, and non-ASCII characters.

Transformers underwent only these basic steps because of their advanced ability to interpret and utilize the contextual meaning of text.

Both Naive Bayes and LSTM models underwent additional preprocessing. This included the removal of contractions, stopwords, punctuation, elongated words, and short words.

For the LSTM, two datasets were tested: one with basic preprocessing, and another with further normalization including spelling corrections and the removal of non-English words. This experiment aimed to determine if more extensive preprocessing could boost LSTM performance.

In the case of Naive Bayes, lemmatization was added to the preprocessing steps to potentially enhance prediction accuracy, as combining proper preprocessing with Naive Bayes classification has been shown to improve results [3].

Although the three models started with the same initial dataset, applying different preprocessing methods resulted in distinctly datasets. To address these variations, two strategies were adopted.

For the test set, it was standardized across all models by selecting only the tweets that, after preprocessing, were common to all models. This approach ensured that the final evaluation of the models' performance was based on a homogeneous test sample, allowing for a fairer and more direct comparison of each model's capabilities.

For the training and validation sets, despite the various preprocessing methods used, it was observed that the total length and class balance of the datasets remained nearly unchanged. Therefore, it was decided to proceed with the analysis without further modifications, retaining the diversity introduced by the specific preprocessing for each model.

For the binary dataset, the approach was to represent each category equitably, particularly focusing on the differentiation between cyberbullying and non-cyberbullying tweets. All tweets that were not related to cyberbullying were grouped to represent the 'non-bullying' class. Conversely, the various types of cyberbullying were grouped together but taken in proportional numbers to form the 'bullying' class. This proportional selection ensured that the numbers of examples in the 'bullying' and 'non-bullying' classes were equal, aiming for a balanced dataset.

System Overview

Naive Bayes Models

In our study, we deployed three distinct variants of Naive Bayes. Each variant was selected based on its unique ability to process and analyze text data.

Multinomial Naive Bayes

The Multinomial Naive Bayes classifier is specifically designed for handling categorical data, such as word/feature counts, which are common in text classification tasks. It operates based on Bayes' theorem to compute the posterior probabilities of classes given feature counts while maintaining the conditional independence assumption among features given the class label. This assumption simplifies the computation of conditional probabilities. The features, represented by word counts, are presumed to follow a multinomial distribution. This means the model considers the frequency of each feature within a class to determine its impact on classifying a document [14].

Complement Naive Bayes

Complement Naive Bayes is typically used for text classification with imbalanced datasets. Unlike traditional Naive Bayes, which calculates the likelihood of an instance belonging to a specific class, Complement Naive Bayes computes the probability that a tweet does not belong to each class. For each class, it evaluates how likely it is that a given tweet falls outside of it [11]. After determining these probabilities for all classes, the class with the lowest probability of non-belonging is chosen. This class selection is based on the premise that the lowest probability of non-belonging indicates the highest likelihood of actual class membership.

Bernoulli Naive Bayes

The Bernoulli Naive Bayes classifier operates on a binary principle that checks whether a term appears in a document or not. When predicting the class of a new tweet, the probabilities of all the features present in the article are multiplied together, as well as the probabilities of non-occurrence for absent features. The class with the highest resultant probability is identified as the most likely class for the tweet [15].

LSTM, Bidirectional LSTM, and Attention Mechanisms

In our study, we employed several variations of the Long Short-Term Memory (LSTM) network. Each LSTM variant was chosen based on its ability to enhance the model's understanding of context in text data.

LSTM (Long Short-Term Memory) Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network particularly well-suited for capturing long-term dependencies in sequential data. Our base model utilizes a standard LSTM network with the following architecture:

- **Embedding Layer**: Converts input words into dense vector representations using pre-trained embeddings.
- **Dropout Layer**: Adds regularization by randomly setting a fraction of input units to zero during training.
- **LSTM Layer**: Processes the sequence data, capturing temporal dependencies.
- **Additional Dropout Layer**

- **Output Layer**: A dense layer with a softmax activation function for multiclass classification or sigmoid activation function for binary classification.

Bidirectional LSTM

To enhance the standard LSTM, we added a bidirectional layer, allowing the model to process the sequence in both forward and backward directions, capturing context from both past and future sequences.

- **Bidirectional LSTM Layer**: Processes the input sequence in both directions, providing a more comprehensive context for each point in the sequence.

Attention Mechanisms

To further improve our model, we integrated a custom attention layer after the LSTM. The attention mechanism allows the model to focus on the most relevant parts of the input sequence for making predictions, which is especially useful in longer sequences.

- **Attention Layer**: Computes attention scores for each time step in the sequence.

Transformer models

To tackle the complex task of identifying cyberbullying content, it was essential to use advanced language models that could understand the context and semantics of the informal texts typical of tweets.
For this reason, we choose to use the following models: BERT, RoBERTa and ELECTRA. In our experiments we included dropout to the hidden layers and attention layers: the latter helped the model to prevent overfitting by randomly setting a fraction of the input units to zero during training; the previous enhanced the model's generalization to new data, ensuring that the attention heads do not become too dependent on specific parts of the input, leading to better generalization [6].

BERT

We decided to use BERT because it has been proven to handle the informal language of tweets effectively and has shown state-of-the-art performance in various NLP tasks [16].
Furthermore, BERT uses word masking to predict hidden words in context, improving its ability to infer the meaning of incomplete or ambiguous sentences, which is common in tweets. In particular, we decided to use two pre-trained models:

- `bert-base-uncased`
- `bert-large-whole-word-masking`

The `bert-base-uncased`, with 110 million parameters and 12 layers, provides a good balance between performance and training speed, making it suitable for many NLP tasks and less resource-intensive. On the other hand, the `bert-large-whole-word-masking`, with 340 million parameters and 24 layers, could potentially lead to better performance due to its deeper and more complex architecture.

Additionally, the *whole word masking* technique in this model can improve the model's comprehension of words in their full context. However, `bert-large-whole-word-masking` requires significantly more computational resources and training time [13].

RoBERTa

RoBERTa, while maintaining the same architecture as BERT, could offer superior contextual understanding and performance due to its optimized training process and the use of larger datasets. These improvements usually make RoBERTa highly effective for NLP tasks such as classification. However, like BERT-large, it requires substantial computational resources.[18]

ELECTRA

ELECTRA, with its innovative *replaced token detection* training method, offers superior efficiency and potentially better performance compared to BERT and RoBERTa. It is highly effective for NLP tasks, requiring fewer computational resources for training compared to BERT-large and RoBERTa, while still maintaining high accuracy and robustness in understanding contextual nuances.[4]

Experiments

Naive Bayes

Our initial experiments focused on Naive Bayes models as a baseline for both multiclass and binary classification problems. We evaluated three variants: Multinomial, Complement, and Bernoulli.

Vectorization Methods

The initial step involved choosing between CountVectorizer or TF-IDF Vectorizer. CountVectorizer operates by counting the frequency of each word appearing in a document, thus creating a matrix of token counts. TF-IDF Vectorizer not only considers the frequency of words but also accounts for their importance across documents. It reduces the weight of commonly used words and boosts the significance of unique terms.

Both vectroizers were evaluated across all features. We compared their performance using accuracy and F1 score on both the multiclass and binary classification tasks. The performance metrics were similar for both vectorizers, with CountVectorizer performing slightly better. Thus, CountVectorizer was chosen as the vectorizer.

Feature Selection

After determining the vectorizer, we moved on to feature selection. We employed the `SelectKBest` method with the Chi-squared (χ^2) statistic as the scoring function. This method prioritizes the selection of the top `k` features that exhibit the strongest correlation with the target variable based on the Chi-squared test results.

We experimented with different levels of feature reduction by setting `k` (the number of features to select) to various percentages of the total features in the training data 10%, 20%, 30%, 50%, 70%, and 100%).

By monitoring the accuracy for each `k` configuration, we aimed to determine the level of feature reduction that yielded the best performance for our classifier. This allowed us to find a balance between model complexity and accuracy.

Ultimately, the `best_k`, the number of features yielding the highest accuracy on the validation set was identified, and each Naive Bayes model was trained and tested with it.

LSTM Networks

As a first step with the LSTM, we investigated the distribution of tweet lengths in the dataset, to establish a threshold for the input sequence length. We observed that only a very small percentage of tweets (approximately 0.04%) were longer than 45 tokens. Consequently, we set 45 tokens as the threshold for tweet length.

Embeddings

The next step involved converting the text into a numeric form using two types of word embeddings: GloVe and FastText.

We downloaded pre-trained GloVe word vectors from the official website. This dataset contains word vectors trained on 2 billion tweets (27 billion tokens, 1.2 million vocabulary, uncased, 1.42 GB download). We used the 50-dimensional and 200-dimensional pre-trained word vectors.

To generate FastText embeddings, we trained a FastText model on our corpus. The model was initialized with the following parameters:

- `min_count = 1`: Increasing `min_count` even slightly (from 1 to 2) drastically reduced the vocabulary size. We decided to set `min_count` to 1 to ensure that all words, even those appearing infrequently, were included in the vocabulary. While this resulted in a lower embedding quality (due to the inclusion of rare words), it allowed us to analyze the LSTM's behavior without the complication of OOV words.

- `vector_size = 100`: This is a common choice for embedding dimension.

- `window = 5`: The context window size was set to 5, after investigating the average tweet length.

- `sg = 1`: We opted for the skip-gram model, which tends to perform better on smaller datasets and is more effective in capturing rare words.

- `epochs = 5`: The default number of training epochs.

- `min_n = 1`: The minimum character n-gram length. This setting ensured that very short words, even those with two characters, were included in the model. This was important, since we observed a high percentage of very short words.

Evaluating embeddings was challenging. We conducted similarity experiments by retrieving the most similar words within and outside the vocabulary. Additionally, we assessed the model's accuracy using the "question word method" (a technique that involves creating analogy questions like "man:woman::king:?" to evaluate the quality of word embeddings). However, given the fact that the multiclass dataset was already limited in size, the resulting accuracy was around 40%, regardless of the level of preprocessing applied. The binary classification task yielded even lower accuracy (less than 5%) due to the even more limited size of its dataset.

After obtaining the embedding matrix, we conducted a randomized grid search over 100 epochs to optimize the hyperparameters of our LSTM model. The search focused on tuning the number of units in the LSTM layer and the dropout rate for the two dropout layerss. Additionally, we decided to experiment by setting the `trainable` attribute to `True` in the embedding layer for the models using GloVe 50 and FastText. Once the most effective hyperparameter combination was identified, we retrained the model with those settings and evaluated its performance on the test set. In both the grid search and retraining, early stopping with a patience of 10 was used to try and prevent overfitting.

Transformers

Following a similar approach used for the LSTM model, we analyzed the distribution of tweet lengths in our dataset to ensure efficient processing and avoid unnecessary computations. This analysis revealed that only a small fraction (approximately 0.04%) of tweets exceeded 70 tokens. Based on this finding, we set a maximum input sequence length of 70 tokens for the Transformer model.

To achieve effective fine-tuning, we employed a grid search approach. In our case, we focused on key hyperparameters like batch size, dropout rates for the attention and hidden layers, and learning rate. We leveraged established practices for Transformer models to guide our selection of hyperparameters for the grid search [12]. Specifically, we explored batch sizes of 32 and 64, finding them to offer a good balance between memory usage and training stability. Similarly, we tested learning rates of 3e-4, 1e-4, 2e-5, 3e-5, 5e-5. However, we excluded 3e-4 based on a previous analysis that indicated significantly poorer performance with this value. Finally, we used dropout probabilities of 0.3 and 0.5 for both the attention and hidden layers.

Given that Transformer models typically require fewer epochs for optimal performance [12], we restricted our grid search to 10 epochs. This ensured efficient exploration of hyperparameter combinations while avoiding unnecessary computational overhead. Additionally, to mitigate the risk of overfitting during training, we implemented early stopping with a patience of 2. Once the optimal combination of hyperparameters was identified through the grid search, each model was re-trained using those hyperparameters. Subsequently, each re-trained model was evaluated on the test set.

Results and Analysis

Considering the balanced nature of the problem, to evaluate our models we opted to utilize both accuracy and F1 score (for binary problems) or Macro F1 score (for multiclass problems). These metrics provide a comprehensive evaluation of the model's performance, considering both overall prediction correctness and the balance between precision and recall across all classes.

Among the different models, Multinomial and Complement Naive Bayes achieve higher accuracy with more features (100% of the features), whereas Bernoulli Naive Bayes performs better with fewer features (around 20-30%). Additionally, Bernoulli Naive Bayes is the best-performing model, reaching an accuracy of 90% in the multiclass problem and 89% in the binary problem.

One possible explanation is that Bernoulli Naive Bayes performs better in cases where word frequency is less important. This is particularly relevant for tweets, which are very short and where words rarely repeat within the same tweet. In such contexts, the mere presence or absence of words is more informative for classification than their frequency. Hence, Bernoulli Naive Bayes, which focuses on binary occurrence, is more effective for this type of text data.

Based on the experiments conducted with LSTM models, we can assert that, in both multiclass and binary classification problems, the models tend to perform better when trained on datasets with less preprocessing. Thus, the additional preprocessing we applied did not lead to significant improvements.

Regarding the different embeddings used, the models generally performed better with GloVe embeddings compared to those created with FastText. This can be explained by the fact that the corpus used to train the FastText embeddings was not extensive. Additionally, tweets often contain informal language, spelling errors, and other issues typical of user-generated content, leading to less effective FastText embeddings in this context. However, the decision to set `trainable=True` in the embedding layer during training likely contributed to the satisfactory results: even the lowest accuracy achieved by a model trained with 100-dimensional FastText embeddings was still 78%.

The variation in model architecture did not lead to significantly different performance outcomes. Generally, in both multiclass and binary cases, models with a bidirectional state tend to performed slightly better, aligning with existing literature where BiLSTM often outperforms regular LSTM. The best performance for the multiclass problem was achieved with a bidirectional LSTM using 50-dimensional GloVe embeddings. For the binary problem, the best performance was achieved with a bidirectional LSTM using 200-dimensional GloVe embeddings.

While all transformer models performed well in classifying ethnicity, age, and religion, the `bert-large-whole-word-masking` model stood out for superior performance on gender and non-cyberbullying categories. It achieved the highest accuracy (0.94) in the multi-class problem, but demands significantly computational resources for training and testing.

For applications with resource constraints, consider `bert-base-uncased` and `ELECTRA`. These models offer a good balance between accuracy (around 0.93) and lower computational demands. `RoBERTa` is another option, although slightly less accurate (0.92). Among transformer models for the binary classification task, bert-base-uncased stood out with an impressive accuracy of 0.93. Notably, all other models also performed remarkably well, consistently achieving accuracy above 0.90.

Model type	**Architecture**	**Accuracy**	**F1 Score**
Naive Bayes	Bernoulli	0.90	0.90
LSTM	Bidirectional	0.93	0.93
Transformers	bert-large-whole-word-masking	0.94	0.94

Table 1: Best models by type - Multiclass

Model type	**Architecture**	**Accuracy**	**F1 Score**
Naive Bayes	Bernoulli	0.89	0.88
LSTM	Bidirectional	0.91	0.91
Transformers	bert-base-uncased	0.93	0.93

Table 2: Best models by type - Binary

Conclusions

Our comparative analysis revealed that all tested models exhibited satisfactory performance, with an unexpected standout performance from the Bernoulli model in both binary and multiclass classification tasks. However, it's noteworthy that BERT emerged as the top-performing model, consistent with existing literature. Despite this success, the challenge of cyberbullying detection remains open.

One of the primary challenges in this task lies in the availability of high-quality, standardized datasets. Addressing this issue is crucial for the development of effective cyberbullying detection systems. The ideal dataset should be balanced, multi-classed, tagged by experts based on clear definitions and rules, and reflect the subtleties of language usage [8].

Moving forward, efforts should focus on improving dataset quality and refining machine learning algorithms to better tackle the nuances of cyberbullying detection. By doing so, we can develop more powerful tools to combat cyberbullying and create safer online spaces for everyone.

References

[1] Aljwharah Alabdulwahab, Mohd Anul Haq, and Mohammed Alshehri. “Cyberbullying Detection using Machine Learning and Deep Learning”. In: *International Journal of Advanced Computer Science and Applications* (2023). URL: http://dx.doi.org/10.14569/IJACSA.2023.0141045.

[2] Rami Belkaroui and Rim Faiz. “Towards Events Tweet Contextualization Using Social Influence Model and Users Conversations”. In: (2015). URL: https://doi.org/10.1145/2797115.2797134.

[3] Priyanga Chandrasekar and Kai Qian. “The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier”. In: (2016). URL: (https://ieeexplore.ieee.org/document/7552291).

[4] Kevin Clark. “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators”. In: (). URL: https://ar5iv.labs.arxiv.org/html/2003.10555.

[5] P. Dalvi et al. “Naive Bayes for Cyberbullying Detection”. In: *IEEE Transactions on Cybernetics* (2020). URL: https://ieeexplore.ieee.org/document/9120893.

[6] Salma El Anigri, Mohammed Himmi, and Abdelhak Mahmoudi. “How BERT’s Dropout Fine-Tuning Affects Text Classification?” In: May 2021, pp. 130–139. ISBN: 978-3-030-76507-1. DOI: 10.1007/978-3-030-76508-8_11.

[7] Arya Fikriansyah. “Improving Text Classification Performance with a Fine-Tuning Approach to Electra Modelsa”. In: (2023). URL: https://medium.com/@aryafikriii/improving-text-classification-performance-with-a-fine-tuning-approach-to-electra-models-f40cbcac252f.

[8] Ali Saffar GEhsan Doostmohammadi Hossein Sameti. “A Deep Word- and Character-based Approach to Offensive Language Identification”. In: (2020). URL: https://arxiv.org/abs/2009.10792.

[9] C. Iwendi, G. Srivastava, and S. Khan. “Cyberbullying detection solutions based on deep learning architectures”. In: *Multimedia Tools and Applications* (2020). URL: https://link.springer.com/article/10.1007/s00530-020-00701-5.

[10] “Keggle dataset”. In: (). URL: https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification/data.

[11] Bhaskar Marapelli, Sreedevi Kadiyala, and Chandra Srinivas Potluri. “Performance Analysis and Classification of Class Imbalanced Dataset Using Complement Naive Bayes Approach”. In: (2023). URL: https://ieeexplore.ieee.org/abstract/document/10083369.

[12] Jack Morris. “Does Model Size Matter? A Comparison of BERT and DistilBERT”. In: (2022). URL: https://wandb.ai/jack-morris/david-vs-goliath/reports/Does-Model-Size-Matter-A-Comparison-of-BERT-and-DistilBERT--VmlldzoxMDUxNzU#:~:text=The%20BERT%20authors%20recommend%20fine,5e%2D5%2C%203e%2D5.

[13] Britney Muller. “BERT 101 State Of The Art NLP Model Explained”. In: (2022). URL: https://huggingface.co/blog/bert-101.

[14] Gurinder Singh et al. “Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification”. In: (2019). URL: https://ieeexplore.ieee.org/abstract/document/8776800?casa_token=4cSdSU8r4GEAAAAA:-xCcas93WbcZ51wLLxmFWD9m_H7BGdgJVz8CZtviFd656QT_21tPeTasBeeQP57u3MtQ.

[15] Gurinder Singh et al. “Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification”. In: (2019). URL: https://ieeexplore.ieee.org/abstract/document/8776800?casa_token=-xR9dgGX6GgAAAAA:3oyNJseZ1KA8Rxh-Lt375zK9PdHlOhDjFaadtpC46enyWLxjksPwbf8414z8Ivc4KvPNFGLNrQ.

[16] Sumit Singh. “BERT Explained: State-of-the-art language model for NLP”. In: (2023). URL: https://www.labellerr.com/blog/bert-explained-state-of-the-art-language-model-for-nlp/.

[17] Teoh Hwai Teng and Kasturi Dewi Varathan. “Cyberbullying Detection in Social Networks: A Comparison Between Machine Learning and Transfer Learning Approaches”. In: *IEEE Access* (2023). URL: https://ieeexplore.ieee.org/document/10122521.

[18] TUNG.M.PHUNG. “A review of pre-trained language models: from BERT, RoBERTa, to ELECTRA, DeBERTa, BigBird, and more”. In: (2021). URL: https://tungmphung.com/a-review-of-pre-trained-language-models-from-bert-roberta-to-electra-deberta-bigbird-and-more/.

Appendix

Data

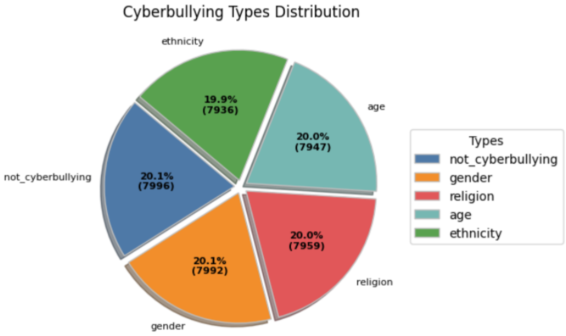


Figure 1: Class Distribution Original Dataset

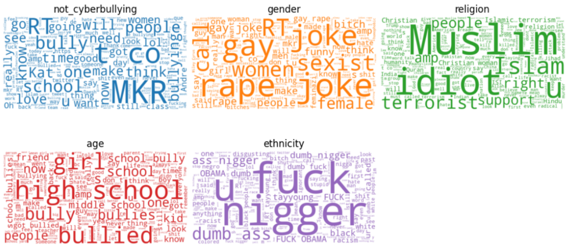
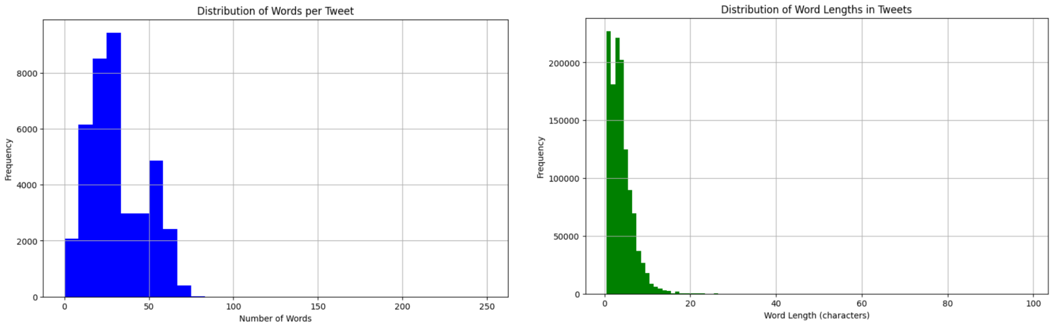


Figure 2: Word Clouds

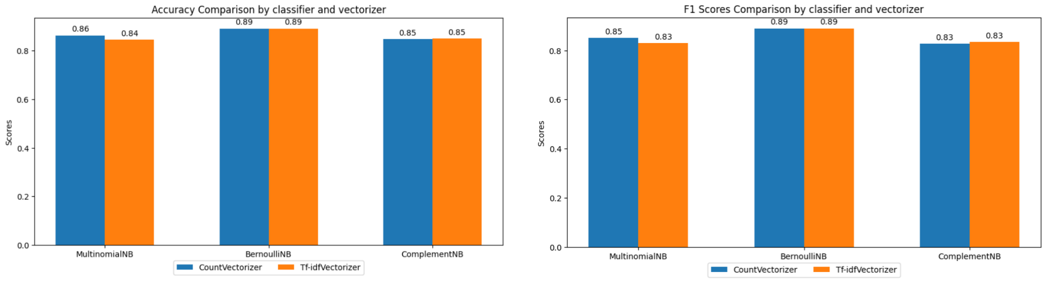


(a)

(b)

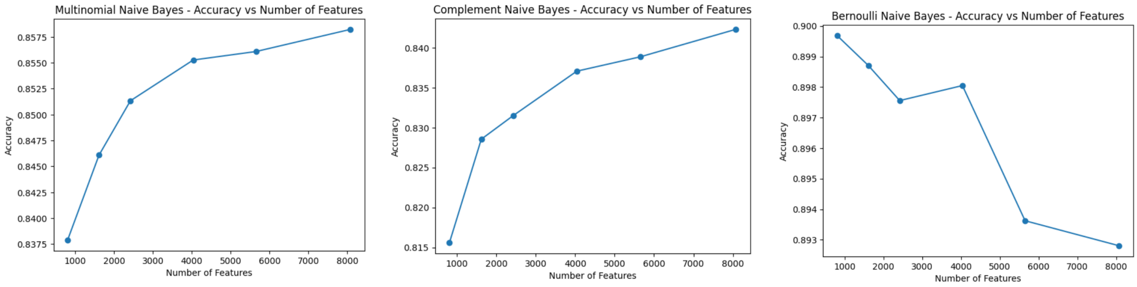
Figure 3: Distribution of words in tweets (a) and of words length (b)

Experiments



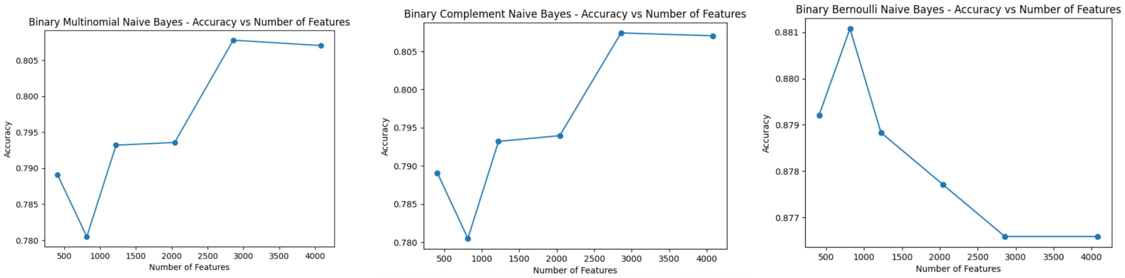
(a) (b)

Figure 4: Comparison between Count Vectorizer and Tf-Idf Vectorizer



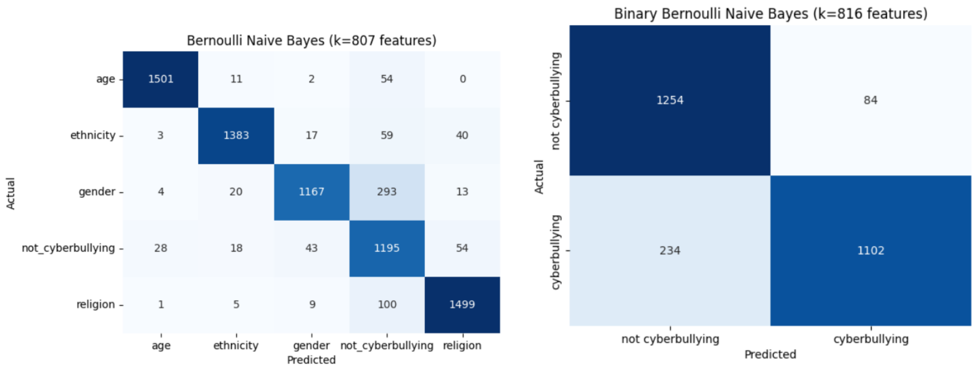
(a) (b) (c)

Figure 5: Multiclass Problem - Feature Selection with Different k Features



(a) (b) (c)

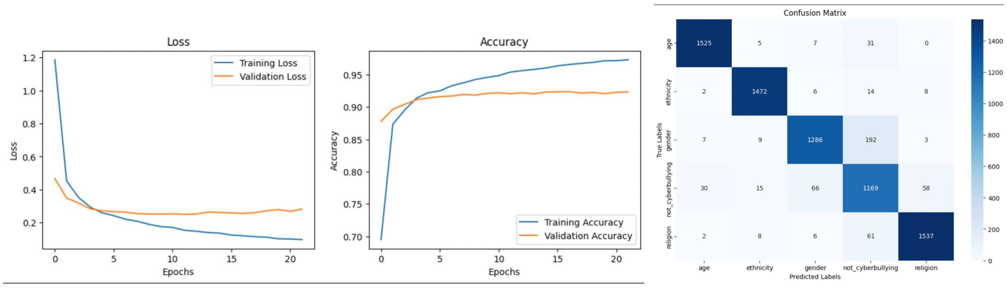
Figure 6: Binary Problem - Feature Selection with Different k Features



(a) Multiclass Problem

(b) Binary Problem

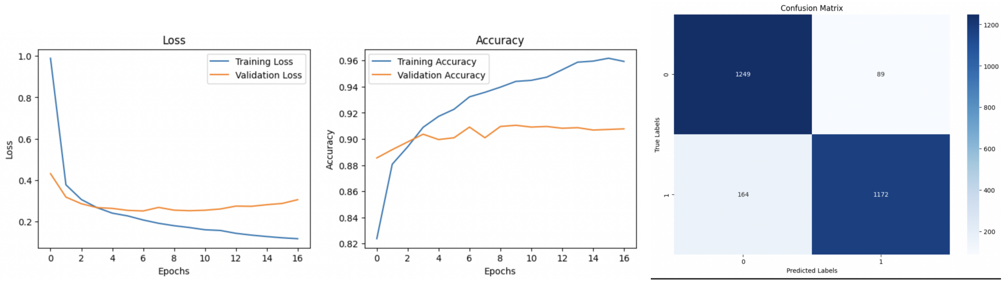
Figure 7: Naive Bayes - Confusion Matrix best models



(a) Loss and accuracy curves

(b) Confusion Matrix

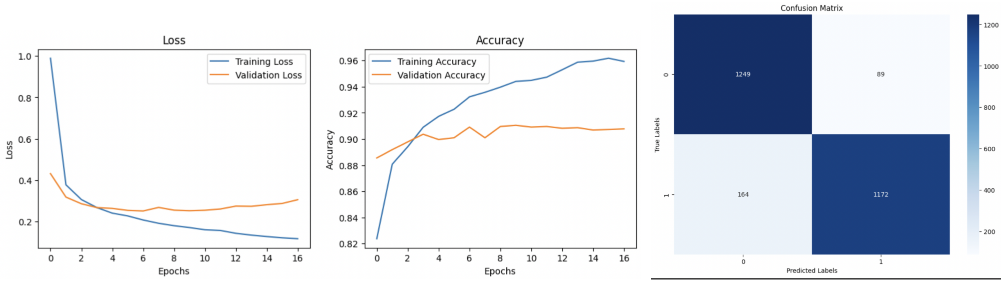
Figure 8: Multiclass Problem: Plot of bidirectional LSTM 50 GloVe embeddings



(a) Loss and accuracy curves

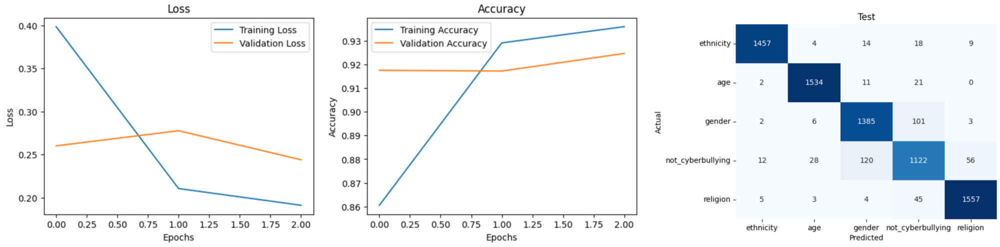
(b) Confusion Matrix

Figure 9: Binary Problem: Plot of bidirectional LSTM 200 GloVe embeddings



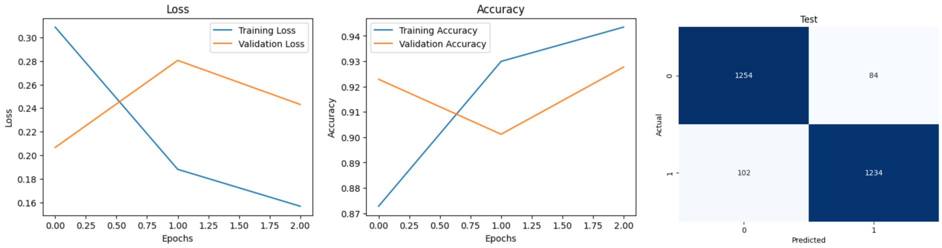
(a) Loss and accuracy curves (b) Confusion Matrix

Figure 10: Binary Problem: Plot of bidirectional LSTM 200 GloVe embeddings



(a) Loss and accuracy curves (b) Confusion Matrix

Figure 11: Multiclass Problem: Plot of bert-large-whole-word-masking



(a) [Loss and accuracy curves (b) Confusion Matrix

Figure 12: Binary Problem: Plot of bert-base-uncased