**Hate Speech Prediction using Supervised and Unsupervised Model**

**Abstract:**

Identifying hate speech on social networks in recent times has attracted considerable interest in the community of natural language processing. This task has been a challenge over years by having a great impact of identifying offensive speech attack on minors, sexuality, ethnic origin, disability, ethnicity, misogyny, and other kinds of hate discriminations. Identifying them as early as possible can have a good impact in real world and can ease the use of social media sites without any harmful contents. Various national and international initiatives have addressed this issue by providing many resources and solutions to the problem. This paper aims to predict hate speech in English language using two methods. Unsupervised K-Means clustering and Supervised deep learning model of Bidirectional Long-Short Term Memory. The unsupervised model uses feature extraction method of TF-IDF and a dimensionality reduction of PCA and trains the model and evaluates on validation set of data and finally tests on a separate new unseen data which predicted F1 score of 57%. The supervised Bi-LSTM training and validation model used word embedding, continuous word representation and other hyperparameter tuning of Epochs, batch size, Learning rate, dimensionality of word embedding and the same was tested on an unseen test data which had the F1 score of 42%.

1. **Materials**

• Codes

* [Supervised Model Training](https://colab.research.google.com/drive/1oZgI-x71HFZpVo9rbPc560DJzfpcHiNU?usp=drive_link%20)
* [Supervised Model Testing](https://colab.research.google.com/drive/1AadRJEOEOHZfT7utFtXiTYvX7UHBNd9S?ouid=118359754890861669399&usp=drive_link)
* [Unsupervised Model Training](https://colab.research.google.com/drive/1lDEcfYQJoGhvai5IHh9TAqrN01f47sIv?usp=drive_link)
* [Unsupervised Model Testing](https://colab.research.google.com/drive/1sLgP2ICEUn9zU_vuToUi-gEcQN9ni6of?usp=drive_link)
* [Google Drive Folder containing code, data, models, and saved outputs.](https://drive.google.com/drive/folders/1g1IDueHfnfMQJjHpB1togTq3sSkyGyy9)

1. **Model Selection**

2.1. Summary of 2 selected Models

The 2 selected models to predict hate speech are,

Bidirectional Long-Short Term Memory

Supervised Model (Bi-LSTM) and K-Means Clustering Unsupervised Model.

Bi-LSTM is a Bidirectional Long Short-Term Memory. It is a variant of the traditional Long Short-Term Memory (LSTM) neural network architecture, which is a type of recurrent neural network (RNN) [1].

Bi-LSTM is specifically designed to capture both past and future context information from input sequences, making it particularly useful for sequence-to-sequence tasks and tasks where understanding the context in both directions is important.

The traditional LSTM is effective in capturing longterm dependencies in a sequence by using a combination of gates (input, output, and forget gates) to control the flow of information. However, it processes the sequence in a

unidirectional manner, meaning it only considers the information from the past while processing each time step.

Bi-LSTM addresses this limitation by introducing a second set of LSTM cells that process the sequence in the reverse direction. During the forward pass, one LSTM processes the sequence from the beginning to the end, while the other LSTM processes the sequence from the end to the beginning. By doing this, the Bi-LSTM can effectively capture information from both past and future contexts.

Each LSTM cell in the forward and backward layers operates independently, but their outputs at each time step are combined to produce the final output of the Bi-LSTM. This combination can be as simple as concatenating the outputs or using more complex techniques like attention mechanisms.

Unsupervised learning is a method to find structure in unlabelled data and for our prediction for hate speech we are making use of **K-Means** clustering model which groups similar instances in the data called as clustering and finds the unusual patterns. Clustering is obtained by calculating the distance between the points of grouped similar data.

Kmeans cluster does its job by grouping similar data points together and discovers its underlying patterns. And to achieve this, K-means looks for a fixed number defined as k of clusters in a dataset. K is where we define the number of centroids which then allocates each data point to the nearest cluster. And the word ‘means’ in K-means refers to averaging the data that finds the centroid. The values of K will be defined by us. Similar data points are grouped into distinct clusters using K-means clustering method depending on how similar their features are. The process begins by converting textual data into numerical representations, such as using word embeddings to capture the semantic meaning of words. These numerical representations serve as feature vectors for each text. Next, K-means is applied, where the algorithm iteratively assigns data points to the nearest cluster centroid based on their feature similarities and recalculates the centroids of each cluster.

The number of clusters (i.e., the number of hate speech classes) needs to be determined beforehand. Once clustering is completed, new text data can be assigned to the cluster with the closest centroid, allowing for the identification of potential hate speech instances.

While K-means clustering can provide valuable insights into the structure of the hate speech data and aid in identifying clusters of similar content, it is important to remember that the generated clusters do not inherently indicate hate speech. Additional human validation and labelling are necessary to verify and interpret the results properly. Moreover, K-means might have limitations when handling high-dimensional or noisy data, and it may struggle to capture more nuanced patterns of hate speech, making it essential to explore other machine learning approaches or combine K-means with other algorithms for more robust hate speech detection systems.

2.2. **Critical discussion and justification of model selection**

**K-Means Clustering:**

**Challenges and Limitations** of K-means clustering model in predicting hate speech.

1. Lack of Semantic Understanding: Kmeans relies on distance-based similarity measures to cluster data. However, hate speech detection requires understanding the semantic meaning and context of words, which K-means may not capture effectively. It treats each feature (word or word embedding) independently and ignores the sequential nature of text, potentially leading to suboptimal clustering results.
2. Selection of K Value: One of the critical challenges with K-means is determining the appropriate number of clusters (k) in advance. Selecting the optimal k value is not always straightforward and can significantly impact the quality of clustering results. Incorrectly chosen k can lead to oversimplification or overcomplication of hate speech categories, affecting the accuracy of the prediction.
3. Ambiguity in Clustering: Hate speech can be multifaceted and diverse, with varying degrees of explicitness. K-means might struggle to distinguish between subtle variations of hate speech or to handle cases where hate speech and non-hate speech content overlap, resulting in ambiguous clustering.
4. Generalization to New Data: K-means does not learn from labeled data, so its clusters may not generalize well to new, unseen hate speech instances. It lacks the ability to adapt to evolving hate speech patterns, making it less suitable for real-time applications.

**Reason for choosing K-Means clustering model:**  Firstly, being an unsupervised learning algorithm, K-means does not require labeled data for training, making it valuable when labeled examples are scarce or unavailable. Secondly, its simplicity and efficiency make it computationally feasible for large datasets and easy to implement. Additionally, K-means generates well-separated clusters, enhancing interpretability, and enabling insightful exploratory data analysis. Its versatility in handling different data types, such as numerical and categorical, expands its applicability across diverse domains. Moreover, the algorithm converges quickly to a solution, contributing to its speed and effectiveness, especially when dealing with sizable datasets. Kmeans also allows for seed initialization, providing users the flexibility to incorporate prior knowledge for better clustering results. It can serve as a baseline for comparing other clustering algorithms or validating the performance of more complex methods. Furthermore, K-means can be utilized as a preprocessing step to group similar data points, reducing complexity in subsequent analysis.

**Bi-Long-Short Term Memory Supervised**  **Learning Model:**

**Bi-LSTM model is well-suited for predicting hate speech due to the following reasons:** BiLSTM's bidirectional nature enables it to extract more informative features from the text, aiding in identifying subtle cues and patterns associated with hate speech.

Hate speech can involve long-range dependencies, and Bi-LSTM is better equipped to capture such dependencies, making it suitable for handling hate speech sequences.

Bi-LSTM captures both past and future context, which is crucial in understanding the context and intent of hate speech, as it often involves complex linguistic patterns and relies on the broader context.

**Limitations and challenges:** While BiLSTM has its advantages it also comes along with certain disadvantages. BiLSTM is a slower model compared to LSTM and requires more time for training.

LSTM models are prone to overfitting, especially when trained on small datasets or with overly complex architectures.

Hate speech detection models can inherit biases present in the training data. Biased data may result in biased predictions, leading to unequal treatment of certain groups or inaccurate identification of hate speech instances. Some hate speech instances can be subtle or ambiguous, requiring deep contextual understanding to distinguish between hate speech and non-hate speech content. LSTM models, while capable of capturing context, may still encounter challenges in disambiguating such instances.

Hate speech instances are often a minority class compared to non-hate speech instances in the dataset. Class imbalance can impact the model's ability to detect hate speech effectively, leading to biased results and reduced sensitivity to hate speech instances.

3. **Design and implementation of classifiers.**

**Dataset representation:**

Below tables represent the label classes for the dataset predicting hate speech.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Total | % Class NOT | % Class OFF |
| Original | 4000 | 91.38 | 8.62 |
| Train | 2800 | 67.32 | 2.67 |
| Valid | 400 | 9.62 | 0.37 |
| Test | 800 | 14.42 | 5.57 |

Table 1: Dataset Details

The original dataset is of 4000 datapoints with 91.38% of Label A (NOT a hate speech) and 8.62% of Label B (OFF indicating as a hate speech) which was later then split into dataset as Train, Validation and Testing as 70, 10, 20 percent, respectively.

As mentioned in the table, the training labels were in the ratio of 67:3 where 67.3% was NOT a hate speech label and 2.6% OFF, meaning, predicted as hate speech.

For the validation set of data which was split in for 10% of which 9.6% came out as NOT a hate

speech and only 0.37% as OFF, yes, a hate speech.

In the Testing dataset which was split for 20%, we had 14.42% of not a hate speech labeled data and 5.57% of hate speech labeled data. It is truly clear that the labeled data are highly imbalanced which also encouraged the split of Train, Valid and Test data to be as highly imbalanced.

Challenges that would be faced when highly imbalanced label data exists are, the model tends to be biased towards majority class as they have more samples to learn from and as a result the model might perform well on majority class but poorly on minority.

Evaluation metrics like accuracies can be misleading in imbalanced datasets. In our case, the model might tend to predict a Hate/Offensive speech as not a hate speech. It might also cause the model to overfit to the minority class by capturing noise rather than learning the meaningful patterns. This will also be difficult for the model to learn the minority class. **Design Implementations:**

For our unsupervised learning model of KMeans clustering we used vectorizer as TF-IDF stands for Term Frequency-Inverse Document Frequency (TF-IDF) and a dimensionality reduction technique named Principal Component Analysis

(PCA).

TF-IDF is a technique used to convert text data into numerical vectors. The Tfidf-Vectorizer function is employed to create a TF-IDF matrix from the preprocessed text data of hate speech. Each row of the matrix represents a document, and each column corresponds to a unique word in the vocabulary. TF-IDF captures the importance of words in a document relative to the entire corpus. Words that are frequent in a specific document but rare in other documents receive higher TF-IDF scores, making them more discriminative for that document.

PCA is a dimensionality reduction technique applied to the TF-IDF matrix. High-dimensional data can lead to increased computational complexity and memory requirements, making it challenging to work with. PCA reduces the number of dimensions while retaining as much information as possible, making subsequent processing more efficient. Overfitting occurs when the model becomes too specific to the training data and fails to generalize to new instances. By reducing dimensionality with PCA, the model becomes less prone to overfitting and may improve its ability to generalize to new data.

Once the model is trained, scores of precisions, recall and F1 along with Silhouette is calculated on both train and validation set of split data. Once the trained model is evaluated on a validation set of data, the model is saved using pickle library along with the vectorizer. This well-trained saved model and vectorizer is then used to make predictions on an unseen data of test split. The results of F1 score is mentioned in the Table 2 and the predicted labels are then added in the output file along with actual test labels and the same is saved as a csv file.

For the supervised model of Bi-LSTM we made use of certain hyperparameter tunings.

The Bi-LSTM model utilizes several important hyperparameters to govern its performance and behaviour during training. Firstly, the vocabulary size parameter determines the size of the vocabulary, representing the total number of unique tokens in the dataset. The embedding dimension hyperparameter sets the dimension of word embeddings, defining the dense vector representation for each word. The number of LSTM units is controlled by lstm units, which affects the hidden state and cell state dimensionality of the bidirectional LSTM layer. During training, the model is exposed to the dataset for a specified number of epochs, and in each iteration, a certain number of samples are processed simultaneously, determined by giving the batch size parameter.

Furthermore, the minimum frequency hyperparameter is employed to filter out tokens with frequencies below a specific threshold from the vocabulary. The learning rate scheduling, using the LR scheduler parameter. Step LR, gradually reduces the learning rate (gamma) after a set number of step size epochs, allowing for smoother convergence during optimization. For loss computation, the model uses the nn.CrossEntropyLoss(), a standard choice for multi-class classification tasks. The model's parameters are updated using the Adam optimizer, an efficient variant of stochastic gradient descent.

The above steps are done on the pre-processed data of both train and validation file and the scores of precisions, recall and F1-score are evaluated. Once the model is trained it is then saved as a pickle file along with the vectorizers that has been included in the training process of the dataset.

This saved model and vectorizer is then used to evaluate the performance and predict the ground truth on an unseen data of test file. Evaluation metrics similar to train and validation is used and the same is mentioned in Table 2. Once the evaluation metric is performed, the output of the predicted label is saved in an output file along with the actual labels.

|  |  |
| --- | --- |
| **Model** | **F1 Score** |
| K-Means Clustering | 56% |
| Bi-LSTM | 42% |
| 1. BERT 2. Bi-LSTM | 96%  95% |

Table 2: Model Performance

1. **Analysis and Discussion**

In the state-of-the-art paper [1] the methods used are Bidirectional Encoder Representations from Transformers (BERT) and Bidirectional Long ShortTerm Memory (Bi-LSTM) which has the F1 scores of 0.96 or 96% and 0.95 or 95% respectively. As a well-known fact, BERT is a pre-trained language model trained on huge data based on contextual representation. [1] BERT is considered as the best state-of-the-art method in most recent language models by providing best results in comparison to other language models for various NLP tasks. Hence, when this model is being used on a new set of data the chances of detecting hate speech would be higher due to its continuous process of training which is in picture over years.

For our Bi-LSTM deep learning supervised model, we trained the model from scratch using a new set of data. The paper’s state-of-the-art method Bi-LSTM scores were about 95% due to its right balance between the labels of data. The default architecture of LSTM is that it saves the data for a long sequence only from left to right, but Bi-LSTM has the advantage of saving the sequences of data from both directions. The Bi-LSTM consists of two LSTMs, one of which analyses the data from left to right and the other in the opposite manner. Both the forward and backward LSTM are then concatenated and flattened to enhance understanding of the surrounding context. This approach has enhanced the evaluation of hate speech prediction to achieve better scores [1] but the same approach in our training model did not perform well.

* 1. **Justification of Model’s performance:**

The dataset used in predicting hate speech in our case had highly imbalanced labelled having a ratio of 97:3 where 97% labels were NOT a hate speech content and only 3% labels were Offensive/Hate speech content. This imbalance of labelled data influenced the machine to learn the major class prediction but unfortunately failed in learning the minority class of labels. The feature extraction methods can be used as continuous bag of words or a skip gram model.

* 1. **Example and other analysis:**

Examples provided in Table 3 from the prediction using our trained model along with the actual labels. Example 1 2 and 3 predicted both Hate speech and Not a hate speech correctly but examples 4 and 5 predictions failed in predicting the hate speech as labelled data. Example 4 was a hate speech, but our model predicted it as not a

hate speech and in Example 5 it was not a hate speech, but our models predicted it as a hate speech. The reason behind this wrong prediction would be that our model trained well on majority class but did not learn well on minority class as the labelled data were highly imbalanced.

1. **Summary** 
   1. **Discussion of work carried out:**

For the unsupervised approach to achieve a better score by performing a hyperparameter tuning technique using Grid Search. Even though we used TF-IDF feature extraction method and PCA dimensionality reduction method we could have alternatively used other feature extraction methods like BoW, BoF1 features, BoF2 features. However, BoF1 features, BoF2 features are outdated since September 2021.

For the deep learning supervised approach, in the paper the hate speech detection using LSTM was based on Italian language which was split into two methods using 10-fold cross validation. In method one the hate speech was split into three categories 1. Strong hate 2. Weak hate 3. No hate and in Method two it was split into binary method of Either a hate speech or Not a hate speech.

For method one, category one of Strong Hate speech scored only 0.097 F1 score whereas the Weak and No hate speech score about 0.221 and 0.747 F1-Scores respectively. In method two, the model predicted a greater number of No hate speech scores and lesser of actual hate speech prediction. Scores of 0.728 of Hate speech and scores of 0.83 of no hate speech.

Even though the scores on predicting hate speech was less, the training of the Bi-LSTM architecture was done by representing each input word by a 262-dimensional vector composed by word embedding, word polarity and End of sentence when indicating whether the word was fully read or not. Using the same approach of feature extraction would have increased the performance of our supervised model a little but there is also a fact that the labels of the dataset were highly imbalanced.

For our supervised model, use of k-fold cross validation could have been used for a better prediction by balancing the labelled data.

**5.2 Lesson Learned:**

The models used could have been better if we had balanced the labelled data and used k-fold cross validation techniques. The predictions were almost as accurate as labelled data but only had a few wrong predictions. For the unsupervised learning model, use of Gridsearch would have a better prediction. Grid search helps in finding an appropriate K value that balances the trade-off between capturing meaningful patterns in the data and avoiding overfitting. It is a crucial step in the K-means clustering process to ensure that the model produces meaningful and interpretable clusters.

|  |  |  |  |
| --- | --- | --- | --- |
| Example | Ground Truth | Supervised | Unsupervised |
| Ex1 - ask user question real | NOT | NOT | NOT |
| Ex2 - cute universe | NOT | NOT | NOT |
| Ex3 - not sand can drown n swim option deep shit get nigger | OFF | OFF | OFF |
| Ex4 - twitter use day disgust really cheating unfollow follower | OFF | NOT | NOT |
| Ex5 - fantastic not can sure wait ooo look user | NOT | OFF | OFF |

Table 3: Comparing two Model’s using 100% data: Sample Examples and model output using Model 1 & 2. GT (Ground Truth) is provided in the test.csv file.

**Bibliographical References**

A. A. a. K. M. Hind Saleha, “Detection of Hate Speech using BERT and Hate Speech Word Embedding with Deep Mode,” Taylor and Francis, 2023.

,. O. F. ,. F. T. I. I. A. O. Femi Emmanuel Ayo, “Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions,” 2022.

https://developer.apple.com/documentation/acc

elerate/bnns/using\_long\_shortterm\_memory\_lay

ers\_lstm,

“developer.apple.com,” *Using Long Short-Term Memory Layers (LSTM).*

P. B. M. d. G. a. G. S. Marco Polignano, “Hate Speech Detection through AlBERTo Italian Language Understanding Model,” 2019.

F. D. F. P. M. S. M. T. Cristina Bosco, “Overview of the EVALITA 2018 Hate Speech Detection Task,” 2018.

S. M. E. C. S. T. S. V. Michele Corazza, “A Multilingual Evaluation for Online Hate Speech Detection”.

D. P. L. Taufic Leonardo Sutejo, “Indonesia Hate Speech Detection Using Deep Learning,” 2018.