Hate speech detection using Twitter datasets

Shaik Rizwana   
*22114611, Data Analytics*  
*National college of Ireland*Dublin, Ireland  
x22114611@student.ncirl.ie

***Abstract* – To understand the concept of homogeneous empirical design when it comes to programming with various libraries to perform similar methods. To understand this case we will leverage “Hate speech detection” project with 3 different datasets.**

# **Context And Objective**

Hate speech has been a growing problem due to the rapid growth of the social media which promotes freedom of speech. Such popular platforms are Twitter, Facebook, Instagram etc. Hate speech is when one targets textual abuse towards a person or a group based on their caste, race, religion, national origin, disability, sex, religion, gender identity and many more sensitive topics[1]

For this project I have identified 3 different datasets with data from Twitter in the form of tweets. The datasets are labelled datasets. The labels include various levels of hate speech like “offensive hate speech” or just normal “hate speech”.

The goal of this project is to identify whether identically named ML techniques perform same when all the parameters are also the same. This is called HOMOGENOUS EMPIRICAL DESIGN.[2]

The project is to observe the answers to the below questions –

1. Do the different implementations of identically named machine learning techniques perform the same? If not, what are the outstanding implementations of the identically named machine learning techniques for specific empirical designs, and evaluation measures? That is, if different implementations of the identically named techniques perform differently, which implementation is better than the others in each dataset? This will identify the best-performing implementation of each[2].
2. How do the datasets employed in this work differ from each other? Specifically, how the three datasets employed are different from each other in terms of their characteristics which can impact the effectiveness of machine learning techniques?[2]

A diagram of data processing

Description automatically generated with medium confidence Figure 1 – Journey of KDD for this project.

# **Datasets**

1. Dataset 1: Hate speech detection using Ethos data[3]

Ethos dataset is a binary labelled dataset for hate speech with respective tweets. This dataset contains 5647 tweets with the following columns with various types of tweets targeting different audiences like disabled, women, immigrants etc. The dataset is clearly labelled by using multiple columns like “sentiment”, ”directness”, “annotator\_sentiment” ,”target” ,”group".

1. Dataset 2: Dynamically generated datasets [4]

This dataset is multilabel and indicates the sentiment, target type of the tweet. It contains 41255 tweets with various columns like "acl.id","X1","text","label","type","target","level","split","round.base","annotator","round","acl.id.matched".

1. Dataset 3: Multilingual hate speech detection against women and immigrants [5]

This dataset contains 1000 tweets that has labels like hate speech, target, aggressive/non-aggressive. The dataset contains the following columns “id”, “text”, “HS”, “TR”, “AG”.

# **Preprocessing**

Since we are implementing a binary classification in machine learning we can have performed multiple levels of textual processing on tweets and one hot encoding on the labels.

For the tweets we have used the below pre-processing steps-

**Textual processing :**

1. Data cleaning – Removing URLs, Removing Emojis, Hashtags, removing mentions, removing punctuation, converting string to lower case, removing numbers and extra whitespaces. This will help to prepare the data for the next steps.
2. Tokenize the text – It is also known as text segmentation/ lexical analysis. In involved a process of breaking down sentences into tokens/words.
3. Remove stop words – After tokenizing the sentences, we process further by removing the stop words that contribute very little meaning to the model like “a”, “an”, “the” etc.
4. Stemming – It is the process of removing affixes or reducing the form of the verbs in order to get the stem(root) word.
5. Lemmatization – NLP or Natural Language Processing is a field that involves computer science, linguistics, and artificial intelligence. It focuses on how computers can understand, analyze, and interpret human language data like text or speech. NLP has various practical applications such as translation, speech recognition, topic segmentation, summarization, and sentiment analysis.
6. TF-IDF - TF-IDF is a method used in natural language processing to determine the significance of a word in a text or a collection of texts. This numerical measure considers the frequency of a word in a text and the frequency of the same word in a collection of texts, and calculates a score accordingly. The more frequently a word appears in a text, the higher its TF-IDF score will be, but if the same word appears frequently across many texts, its score will be offset.

## - Dataset processing :

After the textual processing is done, we can move to the next step where we remove all the unnecessary columns from the datasets. As we are using hate speech label, we are going to drop all the unnecessary columns and rename them into “Tweets” and “labels” for better understanding.

The next step is to tackle the different types of labels present in the datasets. Example, Dataset 1 has sentiment instead of labels, so we are going to make sure all the other labels that are marked “no hate” are encodes as 1 and “no hate” will be encoded as 0. The same has been applied across other two datasets. After this step here are our final datasets with respective columns :

### Dataset 1:

This data set has the following variables –

* Sentiment: It is the intent to hate speech. The different labels present in this column. We need to make sure that all the hate speech intended are marked as 1 and no hate to be marked as 0. Sentiment is also renamed as label to maintain the homogeneity throughout the datasets.

A picture containing text, screenshot, font, algebra

Description automatically generated Figure 2: Top 10 categories in the sentiment label.

* Directness: Classified as direct or indirect hate.
* Annotator\_sentiment: To understand the sentiment behind the hate speech, example: sadness\_disgust, confusion\_indifference, confusion\_indifference etc. This is another label for the data which we will not be using as this is a binary classification cases based on hate speech.
* Target: Label to understand to whom it is targeted towards, example – disability, gender, religion etc.
* Group: It is a breakdown of the target to finer labels, example – special\_needs, women, muslims etc.

As we discussed, the count of the tweets is 5000. The data is highly imbalanced as you can see in the figure 1. By using the method of oversampling, we can see that now the data is balanced between hate speech and no hate speech label.

**A screenshot of a graph

Description automatically generated with medium confidence** Figure 3 : Dataset 1 – Before and after Oversampling

After the pre-processing of the text, below are top 10 common words throughout the dataset one and the frequency of the hate words.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated

Figure 4 : Dataset 1 – Hate words frequency

### Dataset 2:

This dataset has the following variables –

* acl.id': This is a unique identifier for each entry in the dataset.
* 'Text': These are the tweets of the users. This column is later renamed as “tweet” to maintain the homogeneity of the datasets.
* 'Label': This is a variable that indicates whether the content has been classified as hateful or not. The possible values for this variable are 'hate' or 'nothate'. This is easier to perform one hot encoding with.
* 'Type': This is a categorical variable that provides additional information about hateful content. For hateful content, there are five possible values: Animosity, Derogation, Dehumanization, Threatening, and Support for Hateful Entities. For non-hateful content, the type is 'none'. In the first round, the type was not given and is marked as 'notgiven'.
* 'Target': This variable provides information about the group that is being attacked by the hateful content. It can include multiple intersectional characteristics. For non-hateful content, the target is 'none'. In the first round, the target was not given and is marked as 'notgiven'.
* 'Level': This indicates whether the entry is an original piece of content or a modified version of an existing entry.
* 'Round': This is a categorical variable that indicates which round of data entry the entry was added in. The possible values for this variable are 1, 2, 3, or 4, with an 'a' or 'b' appended to indicate whether it is an original entry or a modified version.
* 'Round.base': This variable indicates the round of data entry, but without the appended 'a' or 'b'.
* 'Split': This categorical variable indicates which subset of the dataset the entry belongs to, with the options being 'train', 'dev', or 'test'.
* 'Annotator': This variable indicates which of the 20 annotators entered the content.
* 'acl.id.matched': This is the ID of the corresponding original or modified entry, linking the two versions together. The original ID is given in 'acl.id'.

The count of the tweets is ~41,000 but we have toned it down to 5000 to reduce the complexity with the models. The data is slightly imbalanced as you can see in the below figure. By using the method of oversampling, we can see that now the data is balanced between hate speech and no hate speech label.

A screenshot of a graph

Description automatically generated with low confidence Figure 5 : Dataset 2– Before and after Oversampling

After the pre-processing of the text, below are top 10 common words throughout the dataset one and the frequency of the hate words.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Figure 6 : Dataset 2 – Hate words frequency

### Dataset 3:

The dataset has the following variables –

* “id” – The unique identifier for each tweet.
* “text” – The tweets made. Later renamed as tweet to retain the homogeneity of the datasets.
* “HS” – If Hate speech then 1, else 0.
* “TR” – Encoded 1 if the target is an individual, else 0 for a general group.
* “AG” – If the person who made remarks is aggressive then this is marked 1, else 0.

This dataset contains exactly 1000 tweets. Since the data is unbalanced, I again used oversampling to make sure the models are trained well on the hate and no hate values.

A screenshot of a graph

Description automatically generated with low confidence Figure 7: Dataset 3– Before and after Oversampling

After the pre-processing of the text, below are top 10 common words throughout the dataset one and the frequency of the hate words.

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Figure 8 : Dataset 3 – Hate words frequency

### Word clouds:

To understand the frequency of the data corpus, word cloud plots are really useful. Below are the 3 word clouds for the datasets –

A collage of words

Description automatically generated with low confidence Figure 9 : Word clouds of the Datasets.

# Machine learning techniques – Model building

For this project I have used Naïve bayes and K nearest neighbors(KNN) as the main two algorithm using python libraries. For Naïve bayes I have used “sklearn” and “genism” libraries from python. For KNN I have used “sklearn” and “scipy” libraries from python.

Naïve Bayes : It is a supervised classification learning model. Mainly used text classification. The unique thing about it is that it will not remember the important features that will help the model to differentiate between classes and hence learns from the scratch always.

K – nearest neighbors – This classification method uses distance to make predictions and makes k – clusters of one or more point that are similar in nature.

## Model Building :

### Implimenting Naïve Bayes using “sklearn” and “genism”

Here are the basic parameters that are constant across these two applications –

1. Test and Train split – It is 70% for training and 30% for testing.
2. Random\_State – It is the seed using which training and testing data is divided. To ensure same rows are picked across these implementations, random\_state is 50.

For Naïve bayes below are the hyperparameters considered –

1. Alpha – Alpha is a smoothing parameter that prevents zero probabilities when a feature is not present in the training set.
2. Fit\_prior – It is a Boolean parameter that determines whether to learn prior probabilities or not. In this case it is been set to “True” as default.
3. Class\_prior – Similar to above parameter, it is to adjust the classes according to data or not. In this case it is set to default value “None”.

Below are the hyperparameter values constant according to the datasets –

A picture containing text, screenshot, font, number

Description automatically generated

Figure 10: Hyper-parameters for Naïve Bayes.

### Implimenting KNN using “sklearn” and “scipy”:

Here are the basic parameters that are constant across these two applications –

1. Test and Train split – It is 70% for training and 30% for testing.
2. Random\_State – It is the seed using which training and testing data is divided. To ensure same rows are picked across these implementations, random\_state is 42.

For KNN below are the hyperparameters considered –

1. Algorithm: This parameter specifies the algorithm used to compute the nearest neighbors. The "auto" option automatically selects the most appropriate algorithm based on the training data.
2. Leaf\_size: This parameter controls the size of the leaf node in the KD-tree that is used to store the training data. A smaller leaf size will result in a more accurate but slower query time, while a larger leaf size will result in faster queries but less accuracy.
3. Metric: This parameter specifies the distance metric used to compute the distance between two points in the feature space. The "minkowski" option is a generalization of the Euclidean and Manhattan distance metrics.
4. Metric\_params: This parameter allows additional parameters to be passed to the distance metric function.
5. n\_jobs: This parameter specifies the number of CPU cores to use for parallel processing. A value of "None" indicates that all available cores should be used.
6. n\_neighbors: This parameter specifies the number of nearest neighbors to consider when making a prediction. A larger value of k will result in a smoother decision boundary but may also increase the risk of overfitting.
7. p: This parameter controls the power parameter for the Minkowski distance metric. When p=1, the metric is equivalent to the Manhattan distance, and when p=2, the metric is equivalent to the Euclidean distance.
8. weights: This parameter specifies the weight function used in prediction. The "uniform" option gives equal weight to all neighbors, while the "distance" option gives higher weight to closer neighbors.

A picture containing text, screenshot, font, number

Description automatically generated Figure 11: Hyper-parameters for KNN.

## Results :

Evaluation metrics –

F1 score: It is the score that is a means of precision and recall. F1 score reaches its best value when near to 1 and worst case when near to 0.

Accuracy: It is the measure of how well the model’s prediction is performing.

Below are the Accuracy and F1 score from these methods.

A picture containing text, screenshot, font, number

Description automatically generated Figure 12: Accuracy and F1 score of the datasets.

# Results

Let us now observe the values of the results and see how our experiment has performed.

1. Do the different implementations of identically named machine learning techniques perform the same? If not, what are the outstanding implementations of the identically named machine learning techniques for specific empirical designs, and evaluation measures? That is, if different implementations of the identically named techniques perform differently, which implementation is better than the others in each dataset? This will identify the best-performing implementation of each[2].

No, the identically named machine learning implementation did not perform the same and the values are slightly varying.

**Naïve bayes**: For Dataset 1, sklearn has out-performed genism. For Dataset 2, genism has performed well sklearn and for Dataset 3, Genism performed well.

KNN: For Dataset 1, spicy has outperformed sklearn. For Dataset 2, spicy has performed well than sklearn and for Dataset 3, spicy performed well.

Despite setting up same default hyper-parameters across the two implementations, the results were not same.

This can be a result of the randomization that scikit learn and genism use when implementing certain parts of code. Randomness such has shuffling the data or selecting the data can be different for these codes.

1. How do the datasets employed in this work differ from each other? Specifically, how the three datasets employed are different from each other in terms of their characteristics which can impact the effectiveness of machine learning techniques?[2]

The datasets were completely different as they were picked from different groups of studies. The three datasets were marked multilabel and have different annotations when it comes to hate speech. This can be one of the reason why there is slight variations from 0.1 to 1 in the F1 scores when compared. The same has been described in the “Datasets” section of the report.

# References

[1] K. A. Qureshi and M. Sabih, “Un-Compromised Credibility: Social Media Based Multi-Class Hate Speech Classification for Text,” *IEEE Access*, vol. 9, pp. 109465–109477, 2021, doi: 10.1109/ACCESS.2021.3101977.

[2] Dr Abdul Razzaq, “DATA MINING AND MACHINE LEARNING I National College of Ireland MSc Data Analytics-Year 1 (MSCDAD\_JAN23C\_I) MSc Data Analytics-Year 1 (MSCDAD\_JAN23B\_I) MSc Data Analytics-Year 1 (MSCDAD\_JAN23A\_I).” [Online]. Available: https://docs.google.com/viewer?url=https%3A%2F%2Fcran.r-

[3] I. Mollas, Z. Chrysopoulou, S. Karlos, and G. Tsoumakas, “ETHOS: a multi-label hate speech detection dataset,” *Complex and Intelligent Systems*, vol. 8, no. 6, pp. 4663–4678, Dec. 2022, doi: 10.1007/s40747-021-00608-2.

[4] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, “Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection,” Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.15761

[5] V. Basile *et al.*, “SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter.” [Online]. Available: http://evalita.org