HATE SPEECH DETECTION USING DECISION TREE ALGORITHM

Abstract

The advancement of the internet and social media, people has access to various platforms to f reely share their thoughts and opinions on various topics. However, this freedom of expressio n is abused to incite hatred against individuals or groups of people based on race, religion, ge nder, etc. question. Therefore, to address this emerging problem on social media sites, recent studies have used various feature engineering techniques and machine learning algorithms to automatically detect hate speech posts on different datasets. Advances in machine learning ha ve intrigued researchers seeking and implementing solutions to the problem of hate speech. Currently, we are using decision tree algorithm technique to detect hate speech using text data.

Keywords: groups of people, social media, feature engineering, Decision tree

I. INTRODUCTION

Hate speech can be defined as any speech directed at a group of people on the basis of race, r eligion, ethnicity, nationality, sexual orientation or gender identity. Hate speech is often used to spread hatred and discrimination. It can also be used to intimidate and threaten people. Identifying hate speech is often a job of classifying feelings. Thus, for training, a model capable of classifying hate speech from a specific text snippet can be obtained by training on data typically used to classify sentiment. Therefore, for the task of a hate speech detection model, in this project we used data from Twitter. Hate content on social media continues to rise. While Face book, Twitter, and Google have tried various measures to combat this hateful content, most of them risk violating free speech. Hate speech, on the other hand, offers an effective way to combat online hate without losing freedom of expression.

Therefore, another tactic for these platforms may be to promote hate speech as a means of repelling hateful content. However, to successfully promote this type of hate speech requires a solid understanding of its dynamics in the online world. This understanding is greatly hampered by the lack of well-preserved data. Hate Speech Detection is a model for identifying and tracking hateful and offensive speech on the Internet. Social media is where many people make hateful and offensive comments about others. Therefore, hate speech detection has become an important method for solving problems in today's online world.

Global Internet access has fundamentally changed our view of the world**.** Social media (SM) i s a byproduct of the World Wide Web that comes in many forms: online gaming platforms, d ating apps**,** forums, online news services**,** and social networks. Various social networks target different purposes: dissemination of ideas (Twitter or Facebook), professional contacts (LinkedIn), sharing of images (Instagram), dissemination of videos (YouTube), meetings (Tinder), etc. They all have one thing in common: they try to connect people**.**

The power of social media is such that the number of active users worldwide will reach 3.02 billion per month by 2021. This would represent about a third of the world's population. Among the many social networks in existence, Twitter is currently one of the major platforms and one of the most important data sources for researchers. Twitter is a well-known public web of real-time micro blogs, where news often predates official media. Due to its short message limit (currently 280 characters) and unfiltered feed, its usage has grown rapidly, especially at events, averaging 500 million tweets per day.

II. LITERATURE REVIEW

Hate speech detection using machine learning is an area of research that has received increasing attention in recent years due to the rise of hate speech and its negative impact on society. In this literature review, we look at some of the most important research on using machine learning to detect hate speech.

"Automated Hate Speech Detection and the Offensive Language Problem" by Davidson et al. (2017) This study proposes a deep learning approach to detect hate speech in Twitter posts. The authors trained a convolution neural network (CNN) model on a dataset of tweets labeled as hate speech or not. The results show that the model achieves high accuracy and precision in detecting hate speech.

"Hate Speech Detection on Twitter: A Comparative Study" by Bhardwaj et al. (2019) In this study, the authors compared the performance of different machine learning algorithms, including Naive Bayes, Support Vector Machines (SVM) and Random Forests, in detecting hate speech on Twitter. Waseem and Hovy, "Deep Learning for Hate Speech Detection in Tweets" (2016) This study proposes a deep learning approach to detect hate speech in Twitter posts using a neural network to long- and short-term memory (LSTM). Deshpande and Patel, “Detecting Hate Speech in Social Media Using Ensemble Learning” (2018) In this study, the authors propose an ensemble learning approach for detecting hate speech on Twitter. They combined several machine-learning algorithms, including support vector machines, logistic regression and naive Bayes, to improve model performance. "Hate Speech Detection:

Problem Solved? by Davidson et al. (2017) This article presents a dataset on hate speech and offensive language called the Hate Speech and Offensive Language (HSOL) dataset, which is Neither the collection of tweets is unlabeled. The authors score an F1 score of 0.93 on their data set, showed the effectiveness of their method. "A Hybrid Approach to Hate Speech Detection in Twitter", by Samghabadi and Bagheri (2020) This article proposes a hybrid approach that combines machine learning and rule-based approaches to detect hate speech.

II. METHODOLOGY

A decision tree topic is a representation technique for generating tree structures where each node represents a test on an attribute value and each branch represents the result of the test. The leaves speak to the class. It shows the connections found in the training dataset. This method is quick unless the preparation is very useful. He does not speculate as to the distribution of probabilities for these particular data. The process of constructing a shaft is called inducement.

Building decision trees: The decision tree algorithm is a top-down greedy algorithm, which means producing a tree whose leaves are as uniform as can reasonably be expected. The actual step of the algorithm is to isolate non-uniform sheets into sheets as uniform as expected in this case, until no further splitting is possible. Proposition Systems Interpretability: Decision trees provide a transparent and intuitive way to understand how hate speech detection models make decisions. The tree structure allows the user to walk through the reasoning process and identify the most important features that contribute to the decision.

Speed and efficiency: Decision trees are fast and efficient algorithms for classification tasks, making them suitable for real-time applications. They can handle large datasets with high dimensions and are computationally efficient. Robustness: Decision trees are relatively insensitive to noise and outliers in the data, making them less likely to overfit or underfit the model. This reduces errors and improves model accuracy. Flexibility: Decision trees can be easily modified and adapted to different types of data, making it a versatile tool for hate speech detection.

IV.IMPLEMENTATION

Steps in building Hate Speech detection using Decision Tree

Before jumping straight to the implementation part, let's dive into the steps involved in building a hate speech detection project using Python.

- Set up the development environment
- Understand the data
- Import the required libraries
- Preprocess the data
- Splitting the data
- Construct the model
- Measure the results

1. **Setting up the development environment**

The first step is to set up a development environment to create a hate speech detection project using Python. To develop a hate speech detection project, you must have jupyter notebook software installed on your system. Otherwise, you can also use Google Collab https://colab.research.google.com/ for the development of the project.

2. **Understanding the data**

The dataset used to build our hate speech detection model is available at www.kaggle.com. This dataset consists of Twitter hate speech detection data and is used to study hate speech detection. The text in the data was categorized as hate speech, offensive language, and neither.

Given the nature of the research, it is important to note that this dataset includes texts that can be considered racist, sexist, homophobic or generally offensive. The Hate Speech Detection dataset is comprised of 7 columns that include the index, count, hate speech, offensive language, none, class and tweet.

Index - This column has an index number.
Count - Indicates how many users coded each tweet.
Hate Speech- This column lists the number of users who have tagged a tweet as hateful.
Offensive Language – There is the number of users who rated the tweet as offensive.
Neither – This indicates the number of users who rated the tweet as neither. Offensive nor non-offensive
Class – It has a class label for the majority of users, where 0 signifies hate speech, 1 signifies offensive language and 2 does not designate either.
Tweet – The following column contains the text of the tweet.

3. **Importing the required libraries**

After analyzing the data, our next step is to import the libraries needed for the project. A few libraries we use in this project are pandas, numpy, scikit learn, and nltk.

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

fig 1: Importing libraries

Let's import the NLTK (The Natural Language Toolkit) library, used for the symbolic and statistical processing of natural language for English written in Python programming language.

```
import nltk
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
import re
import nltk
from nltk.util import pr
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword = set(stopwords.words("english"))
```

fig 2: Natural language Processing

Once the necessary libraries have been imported, it is time to upload the data to our project.

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving twitter_data.csv.csv to twitter_data.csv.csv
```

```
import io
df=pd.read_csv(io.BytesIO(uploaded['twitter_data.csv.csv']))
print(df.head())
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet
0  !!! RT @mayasolovely: As a woman you shouldn't...
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
```

fig 3: Data upload

```
import seaborn as sns
import matplotlib.pyplot as plt
graph = sns.FacetGrid(data=df, col='class')
graph.map(plt.hist, 'text length', bins=50)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0bf16c9fd0>
```

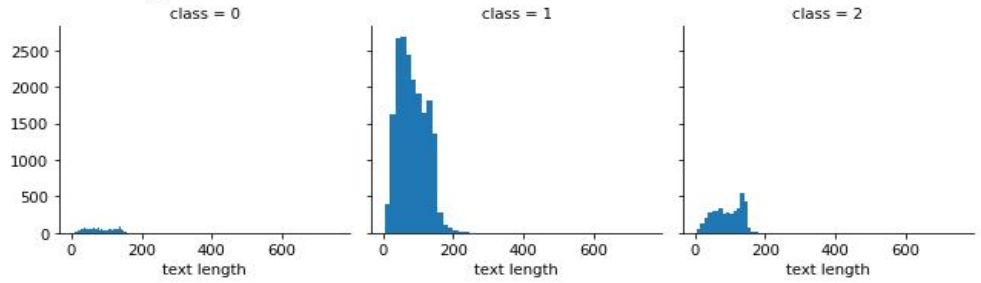

fig 4: Distribution of text length seems

- The distribution of text length seems almost the same in all three classes.
- The number of tweets seems to be a lot higher in the direction of class 1.

Box-plot visualization

```
[ ]  sns.boxplot(x='class', y='text length', data=df)
```

```
<Axes: xlabel='class', ylabel='text length'>
```

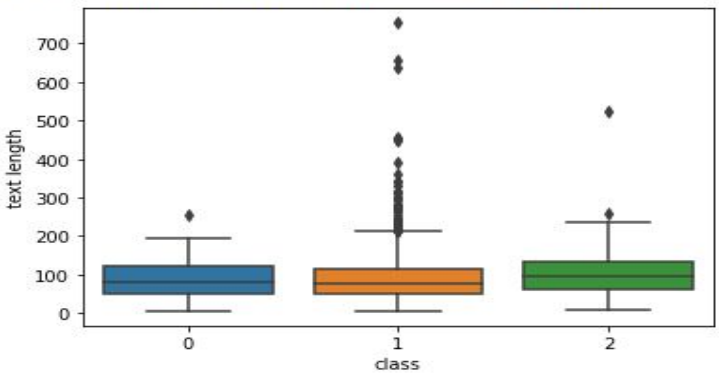

fig 5: Box-plot visualization

From the box-plot, it seems that category 1 tweets have a much longer text. There are also outliers present such that the length of the text will not be a useful feature to consider.

```
df['class'].hist()
```

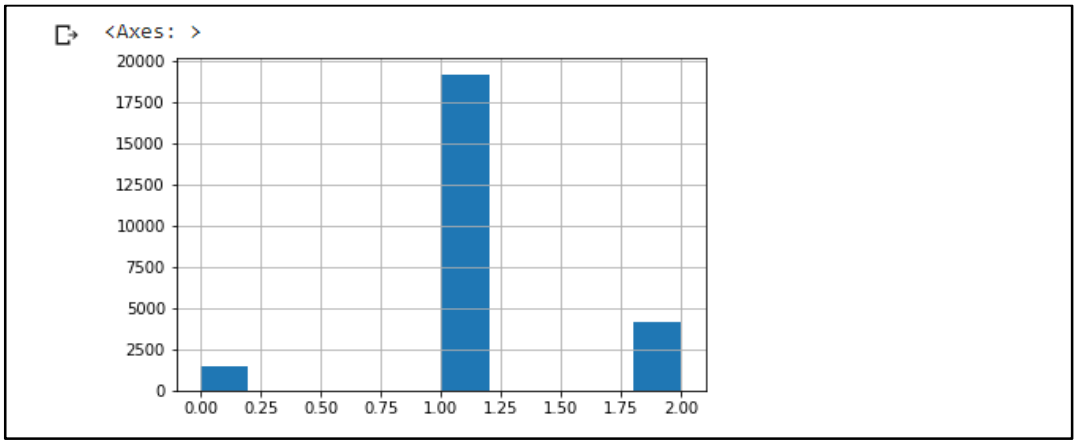

fig 6: histogram showing offensive words by CF coders

The histogram above shows that most tweets are regarded as offensive words by CF coders.

4. Preprocessing the data

As part of the initial data processing, we prepare the raw data and compare it with an automated learning model. This is a crucial first stage in the development of a ML model. When you create an ML project.

```
df['labels'] = df['class'].map({0:"Hate Speech Detected",1:"Offencive language detected",2:"No hate and  No offensive speech"})
print(df.head())
```

```
   Unnamed: 0  count  hate_speech  offensive_language  neither  class  \
0           0      3            0                   0        3      2
1           1      3            0                   3        0      1
2           2      3            0                   3        0      1
3           3      3            0                   2        1      1
4           4      6            0                   6        0      1

                                               tweet  text length  \
0  !!! RT @mayasolovely: As a woman you shouldn't...          140
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...           85
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...          120
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...           62
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...          137

                          labels
0  No hate and  No offensive speech
1       Offencive language detected
2       Offencive language detected
3       Offencive language detected
4       Offencive language detected
```

fig 7: Preprocessing the data

We may not find data that is clear and formatted. Before doing anything with the data, it should be cleansed and formatted. To do that, we use data preprocessing tasks.

```
df=df[['tweet','labels']]
df.head()
```

	tweet	labels
0	!!! RT @mayasolovely: As a woman you shouldn't...	No hate and No offensive speech
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	Offencive language detected
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	Offencive language detected
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	Offencive language detected
4	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	Offencive language detected

fig 8: data preprocessing tasks

We use two important terms for natural language processing, stop words and stems. Stop words are words (data) that are not useful to deal with natural language. We can avoid typing these words. Stemming is the process of creating morphological variations of stem words. We need to find better and more predictable roots for each text

```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]','',text)
    text = re. sub('https?://\S+|www\.\s+','',text)
    text = re.sub('<.?>+','',text)
    text = re.sub('[%s]'% re.escape (string.punctuation),'',text)
    text = re. sub('\n','',text)
    text = re.sub('\w*\d\w*','',text)
    text = [word for word in text.split(' ') if word not in stopword]
    text = " ".join(text)
    text = [stemmer. stem(word) for word in text.split(' ')]
    text = " ".join(text)
    return text

df ["tweet"] = df ["tweet"].apply(clean)
print(df.head())
```

```
                                               tweet  \
0   rt mayasolov woman shouldnt complain clean ho...
1   rt  boy dat coldtyga dwn bad cuffin dat hoe  ...
2   rt urkindofbrand dawg rt  ever fuck bitch sta...
3            rt cganderson vivabas look like tranni
4   rt shenikarobert shit hear might true might f...

                             labels
0  No hate and  No offensive speech
1       Offencive language detected
2       Offencive language detected
3       Offencive language detected
4       Offencive language detected
```

fig 9: predictable roots for each text

5. Splitting the data

The next important step is to explore the dataset and split it into training and test data.

```
[ ] x = np.array(df["tweet"])
    y = np.array(df["labels"])

    cv = CountVectorizer()
    X = cv.fit_transform(x)
    X_train,X_test,y_train,y_test = train_test_split(X,y, test_size= 0.33, random_state= 42)
```

fig 10: Splitting the data

6. Construct the model

After data separation, our next task is to find the right algorithm for our model. A decision tree classifier can be used to construct a project to detect hate speech. Decision trees are a type of supervised machine learning primarily used for classification problems.

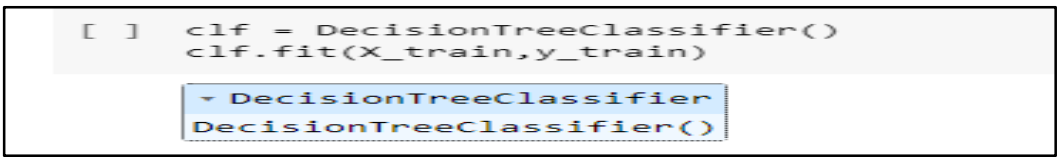

fig 11: Model construction

7. Measure the results

The final step in the development of a MLmodel is prediction. In this step, we can measure th e performance of our model on the test items.

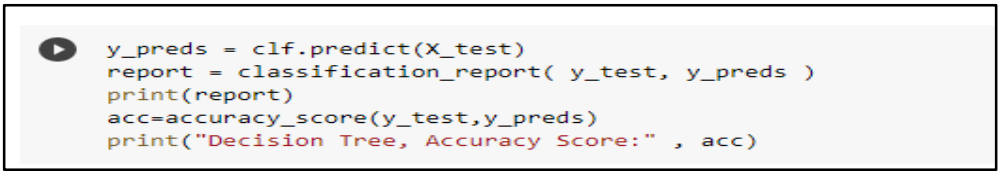

fig 12: Measure the results

Output:

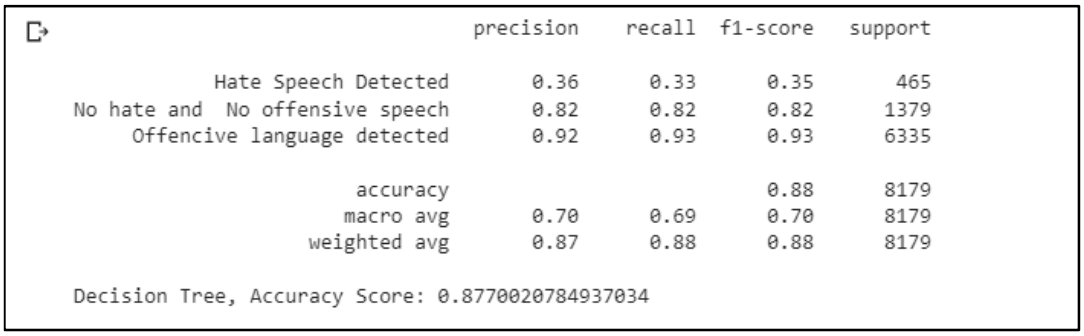

fig 13: Output of work

We can conclude that our hate speech detection model is 87% accurate**.**

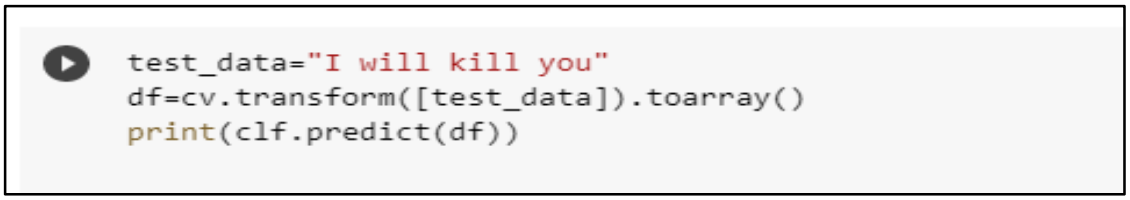

fig 14. speech detection model

Output:

```
['Hate Speech Detected']
```

```
test_data="you are awesome"
df=cv.transform([test_data]).toarray()
print(clf.predict(df))
```

fig 15: hate speech test data

Output:

```
['No hate and  No offensive speech']
```

```
test_data="You are too bad and I dont like your attitude"
df=cv.transform([test_data]).toarray()
print(clf.predict(df))
```

fig 16: Non hate speech test data

Output:

```
['Offencive language detected']
```

V. Conclusion

Researchers have a long history of ignoring hate and offensive language. In this project, we are applying a decision tree algorithm to identify hate language on social media platforms. We find that our model performs remarkably well compared to previous models. We built a hate speech detection project using decision trees. Hate speech is a major issue that we see on social media platforms like Facebook and Twitter, and we can detect it with the most specific models.

References

[1]https://thesai.org/Downloads/Volume11No8/Paper_61-Automatic_Hate_Speech_Detection.pdf

[2] https://www.irjet.net/archives/V9/i6/IRJET-V9I6671.pdf

[3] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Detecting offensive language in tweets using deep learning," arXiv preprint arXiv:1801.04433, 2018.

[4] F. Del Vigna12, A. Cimino23, F. Dell'Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook," 2017.

[5] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Eleventh International AAAI Conference on Web and Social Media, 2017.

[6] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in Proceedings of the 25th international conference on world wide web. International World Wide Web Conferences Steering Committee, 2016, pp. 145–153.

[7] W. Warner and J. Hirschberg, "Detecting hate speech on the world wide web," in Proceedings of the second workshop on language in social media. Association for computational Linguistics, 2012, pp.19-26.

[8] A. Heydarzadegan et al, "Evaluation of Machine Learning Algorithms in Artificial Intelligence", International Journal of Computer Science and Mobile Computing (IJCSMC), Vol.4 Issue.5, pg. 278-286, May 2015.

[9] A. Navlani, "Understanding Random Forests Classifiers in Python", 2019. [Online].Available at: https://www.datacamp.com/community/tutorials/random-forests-classifi er-python. [Accessed: 10 April 2019].

[10] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", Nov 2018, arXiv:1808.03314v4 [17] "Recurrent Neural Networks", 2019. [Online]. Available at: https://machinelearning-blog.com/2018/02/21/recurrent-neural-networks [Accessed: 10 April 2019].

[11] C. Goutte, É. Gaussier. "A Probabilistic Interpretation of Precision Recall and F-

Score, with Implication for Evaluation", ECIR, 2015. [19] R. Joshi, "Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures", 2016. [Online]. Available at: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretat ion-of-performance-measures/. [Accessed: 10 April 2019].

[12] S. Ahammed, M. Rahman, M. H. Niloy and S. M. M. H. Chowdhury, "Implementation of Machine Learning to Detect Hate Speech in Bangla Language," 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), 2019, pp. 317-320, doi: 10.1109/SMART46866.2019.9117214.

[13] R. K. Giri, S. C. Gupta and U. K. Gupta, "An approach to detect offence inMemes using Natural Language Processing (NLP) and Deep learning," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-5, doi: n10.1109/ICCCI50826.2021.9402406.