

TOXIC COMMENTS CLASSIFICATION USING NLP

Dr. M. V. Vijaya Saradhi¹, Konda Yuva Kumar², Banoth Bhaskar³, Udayagiri Sai Charan⁴,
Boggula Kharthik⁵

¹Professor And Head, CSE and Iot Dept Ace Engineering College Hyderabad, India.

^{2,3,4,5}CSE Ace Engineering College Hyderabad, India.

DOI: <https://www.doi.org/10.58257/IJPREMS33180>

ABSTRACT

"Building a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insult and identity-based hate. Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on

seeking different opinions. Platforms struggle to efficiently facilitate conversations, leading many communities to limit or completely shut down user comments. So far we have a range of publicly available models served through the perspective APIs, including toxicity. But the current models still make errors, and they don't allow users to select which type of toxicity they're interested in finding Online discussions often face challenges due to toxic comments, like threats, obscenity, insults, and identity-based hate. Existing models and APIs can help identify toxic content but often lack precision and user customization.

They might make mistakes, discouraging people from participating in discussions. To improve online conversations, we need smarter models that can accurately spot and categorize different types of toxicity. These models should allow users to select what kind of harmful content they want to filter, making online interactions safer and more open. This way, we can create a more welcoming digital space where people can freely express their thoughts and ideas."

1. INTRODUCTION

Toxic Comments Classification Using Natural Language Processing (NLP) is a critical application within the field of machine learning and artificial intelligence. In the digital age, online platforms often face challenges related to toxic or abusive comments, which can have detrimental effects on user experiences, online communities, and brand reputation. NLP, as a subfield of AI, enables the development of models and algorithms capable of automatically identifying and categorizing toxic comments in textual data. The goal of Toxic Comments Classification is to create robust and accurate models that can distinguish between different types of harmful language, such as hate speech, offensive language, or threats, within user-generated content. This process involves training machine learning models on labeled datasets, where comments are annotated based on their toxicity levels. The models learn patterns and features from these examples, allowing them to generalize and classify unseen comments effectively.

The significance of this application extends beyond content moderation. Platforms that host user-generated content, social media, news websites, and online forums, for example, can use Toxic Comments Classification to maintain a healthier online environment, fostering positive interactions and reducing the risk of harm.

Key components of a Toxic Comments Classification system using NLP include data preprocessing, feature extraction, model training, and evaluation. Various NLP techniques, such as tokenization, stemming, and sentiment analysis, are employed to enhance the model's understanding of context and language nuances. As the field of NLP continues to advance, leveraging deep learning techniques like recurrent neural networks (RNNs) and transformers has become common in developing more sophisticated and accurate toxic comment classifiers.

2. OBJECTIVES

In the context of toxic comments classification using NLP, the primary objectives include developing an accurate NLP model to identify and categorize toxic comments and enhancing user experience through customization. The project aims to improve model accuracy, ensure scalability, provide multilingual support, and address ethical considerations to create safer online spaces.

the objectives revolve around creating an accurate, customizable, and adaptable NLP model for toxic comments classification, with a focus on user empowerment, ethical considerations, and community engagement to foster a more inclusive and respectful online environment. Domain: Machine Learning and Artificial Intelligence , Text Processing, Natural Language Processing(NLP) Technologies required: Text Preprocessing, Tensorflow, NumPy, NLTK, pyTorch google cloud.

3. PROBLEM STATEMENT

In the age of online communication, the rise of toxic comments, including threats, obscenity, insults, and identity-based hate, has become a significant challenge, hindering constructive and respectful discussions. The objective is to develop a natural language processing (NLP) solution that can accurately and efficiently identify and categorize various forms of toxicity in user-generated content. This solution should empower users to customize their content filtering preferences to enhance their online experience, promote more meaningful conversations, and create a safer and more inclusive digital environment.

In the realm of online communication, the challenge is identifying and classifying toxic comments, including threats, obscenities, insults, and identity-based hate. Our goal is to develop a precise NLP solution that allows users to customize their content filtering preferences, creating a safer, more inclusive digital environment. This project aims to improve online discourse quality, reduce false positives and negatives, and mitigate the harmful impact of abuse and harassment.

4. PROPOSED SYSTEM

The proposed system for toxic comments classification using NLP integrates advanced features to overcome existing challenges. It emphasizes context-aware models to comprehend nuances and cultural variations, ensuring accurate classification. Continuous learning mechanisms are implemented to adapt to evolving language trends, incorporating new slang and expressions. Bias mitigation techniques, including debiasing algorithms, address and rectify biases present in training data for fair and unbiased classifications. Dynamic datasets, spanning various linguistic styles and cultural backgrounds, enhance the model's ability to generalize effectively. The system employs a human-in-the-loop validation approach, allowing users to provide feedback, ensuring ethical considerations, and refining model outputs. These strategies collectively aim to create a robust and adaptable toxic comments classification system that aligns with evolving communication dynamics while prioritizing accuracy and fairness.

5. HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS:

- Processor – Pentium IV
- RAM – 4 GB (min)
- Hard Disk – 20 GB
- Key Board – Standard Windows Keyboard
- Mouse – Two or Three Button Mouse
- Monitor – SVGA

SOFTWARE REQUIREMENTS:

- Operating system – Windows 7, 8, 10, 11, mac os
- Coding Language – Python
- Back-End – Python
- Designing – HTML , CSS , Java Script

6. TECHNOLOGY DESCRIPTION PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

7. PACKAGES USED

Flask is a lightweight web framework for Python, designed for simplicity and flexibility in web application development. Developed by Armin Ronacher, it utilizes the Werkzeug WSGI toolkit and Jinja2 template engine. Key features include a minimalist structure, decorator-based routing, a built-in development server, and integration with Jinja2 for templating. Flask's modular design allows developers to select and integrate components as needed, and its

rich ecosystem of extensions facilitates the addition of features like authentication and database integration. Installation is straightforward using pip, and a typical "Hello World" example showcases its ease of use. While Flask is flexible in project structure, a common convention involves organizing applications with a package structure. Deploying Flask applications can be done on various platforms, from traditional web servers to cloud services. Overall, Flask's simplicity, flexibility, and extensive documentation make it a popular choice for Python web development.



ALGORITHM

Word NetL emmatizer:

The Word NetL emmatizer is a tool provided by NLTK (Natural Language Toolkit) for lemmatization in natural language processing. Lemmatization involves reducing words to their base or root form, known as the lemma. This process helps in standardizing and normalizing words, making it useful for tasks like text analysis, information retrieval, and machine learning.

TfidfVectorizer:

The TfidfVectorizer is a feature extraction method commonly used in natural language processing and information retrieval. It transforms a collection of raw text documents into a matrix of TF-IDF features. TF-IDF stands for Term Frequency-Inverse Document Frequency, and it reflects the importance of a term in a document relative to its frequency across multiple documents. TfidfVectorizer is widely employed in text-based machine learning tasks, such as document classification and clustering, where it helps in capturing the significance of words within a corpus.

MultinomialNB (Multinomial Naive Bayes):

MultinomialNB is a classification algorithm based on the Naive Bayes theorem, specifically designed for datasets with discrete features. It is commonly used in text classification tasks, including spam filtering, sentiment analysis, and document categorization.

The "naive" assumption is that features are conditionally independent, simplifying the computation of probabilities. MultinomialNB is particularly effective with features representing word counts or term frequencies, making it well-suited for natural language processing applications. It has proven to be a robust and efficient choice for text classification problems.

Coding:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
data = pd.read_csv('FinalBalancedDataset.csv')
data.info()
data.head(5)
data = data.drop("Unnamed: 0", axis=1)
data.head(5)
data['Toxicity'].value_counts()
import nltk
nltk.download('punkt')
```

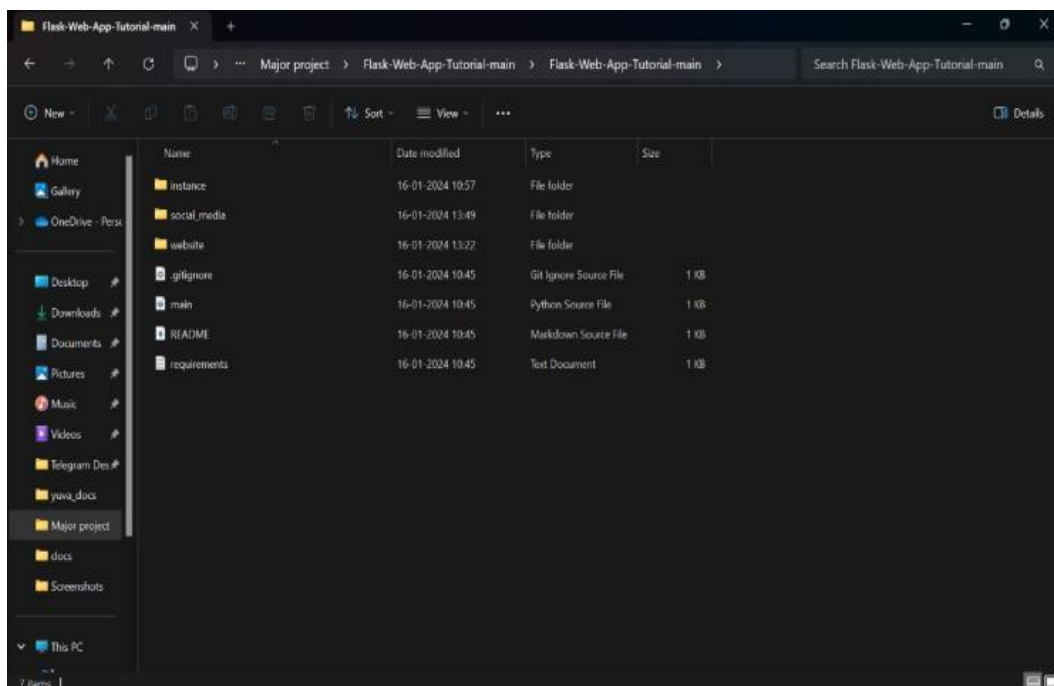
```
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
from nltk import WordNetLemmatizer
from nltk import pos_tag, word_tokenize
from nltk.corpus import stopwords as nltk_stopwords
from nltk.corpus import wordnet
## Text pre-processing
wordnet_lemmatizer = WordNetLemmatizer()
import re
def prepare_text(text):
def get_wordnet_pos(treebank_tag):
if treebank_tag.startswith('J'):
return wordnet.ADJ
elif treebank_tag.startswith('V'):
return wordnet.VERB
elif treebank_tag.startswith('N'):
return wordnet.NOUN
elif treebank_tag.startswith('R'):
return wordnet.ADV
else:
return wordnet.NOUN
text = re.sub(r'[^a-zA-Z\']', ' ', text)
text = text.split()
text = ' '.join(text)
text = word_tokenize(text)
text = pos_tag(text)
lemma = []
for i in text: lemma.append(wordnet_lemmatizer.lemmatize(i[0], pos = get_wordnet_pos(i[1])))
lemma = ' '.join(lemma)
return lemma
data['clean_tweets'] = data['tweet'].apply(lambda x: prepare_text(x))
data.head(5)
## Tfidf for features
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
corpus = data['clean_tweets'].values.astype('U')
stopwords = list(set(nltk_stopwords.words('english')))
count_tf_idf = TfidfVectorizer(stop_words = stopwords)
tf_idf = count_tf_idf.fit_transform(corpus)
import pickle
pickle.dump(count_tf_idf, open("tf_idf.pkl", "wb"))
tf_idf_train, tf_idf_test, target_train, target_test = train_test_split(
```

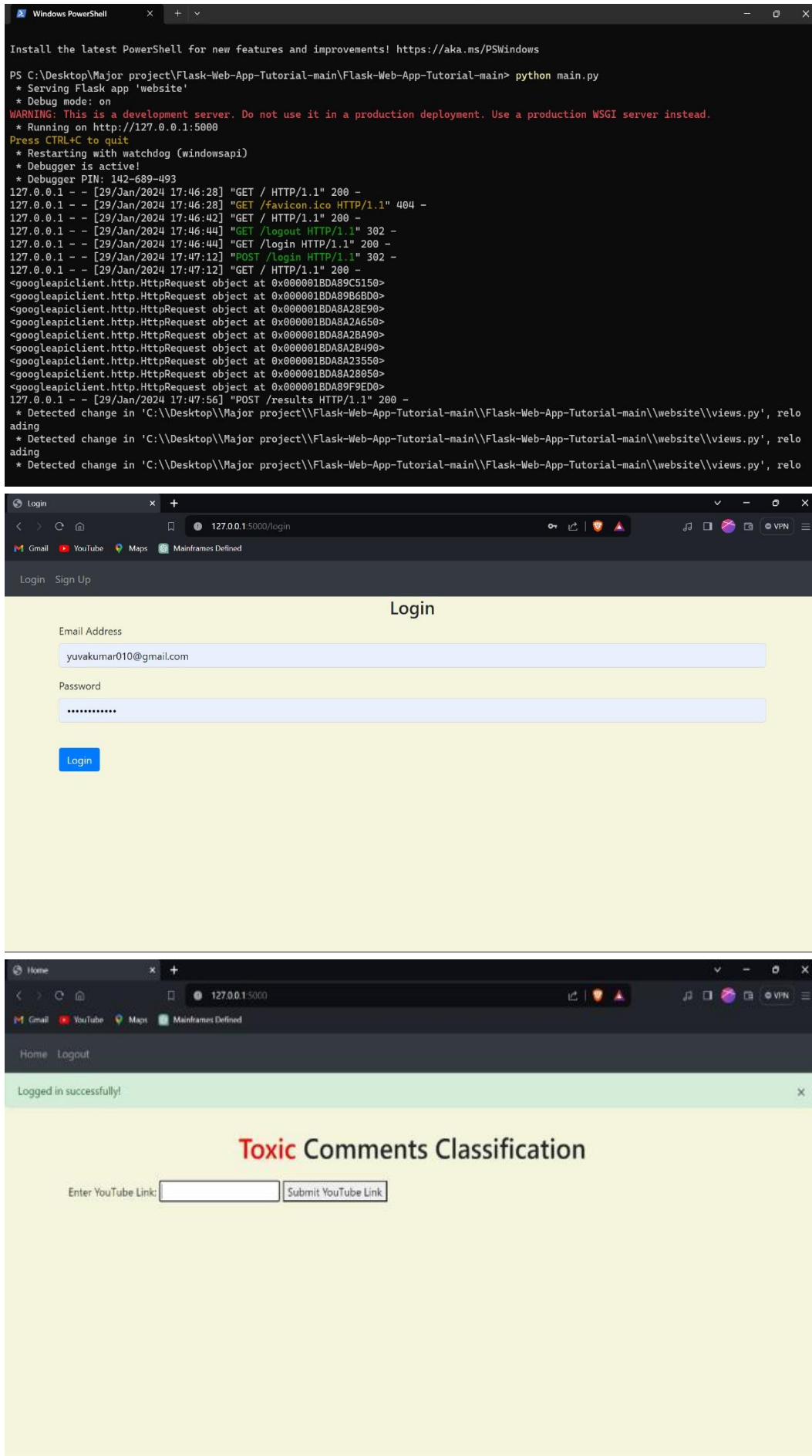
```
tf_idf, data['Toxicity'], test_size = 0.8, random_state= 42, shuffle=True
)
## Create a Binary Classification Model
model_bayes = MultinomialNB()
model_bayes = model_bayes.fit(tf_idf_train, target_train)
y_pred_proba = model_bayes.predict_proba(tf_idf_test)[::, 1]
y_pred_proba
fpr, tpr, _ = roc_curve(target_test, y_pred_proba)
final_roc_auc = roc_auc_score(target_test, y_pred_proba)
final_roc_auc
test_text = "I am gone kill you"
test_tfidf = count_tf_idf.transform([test_text])
display(model_bayes.predict_proba(test_tfidf))
display(model_bayes.predict(test_tfidf))
## Save the model
pickle.dump(model_bayes, open("toxicity_model.pkt", "wb"))
model=pickle.load(open("toxicity_model.pkt","rb"))
model
output=model.predict(count_tf_idf.transform(["good"]))
"Toxic" if output[0]==1 else "Non-Toxic"
import numpy as np
import pandas as pd
# import streamlit as st
from googleapiclient.discovery import build
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
import re
import os
# youtube comments extraction process
# Replace 'YOUR_API_KEY' with the API key you obtained
API_KEY = 'AIzaSyADZ3xWnP1_RBK68r7ADwKSAEY9PgxEY5E'
youtube = build('youtube', 'v3', developerKey=API_KEY)
comment=[]
authorname=[]
toxic=[]
def extract_video_id(video_url):
# Regular expression to match the YouTube video ID
pattern = re.compile(r'(?:(?:youtube\.com)/(?:[^\n\s]+/\S+/(?:v|e(?:mbed)?)/\S*?[?&]v=)|youtu\.be/)([a-zA-Z0-9_-]{11})')
# Search for the pattern in the URL
match = pattern.search(video_url)
# If a match is found, return the video ID
if match:
return match.group(1)
else:
return "No id found! sorry"
```

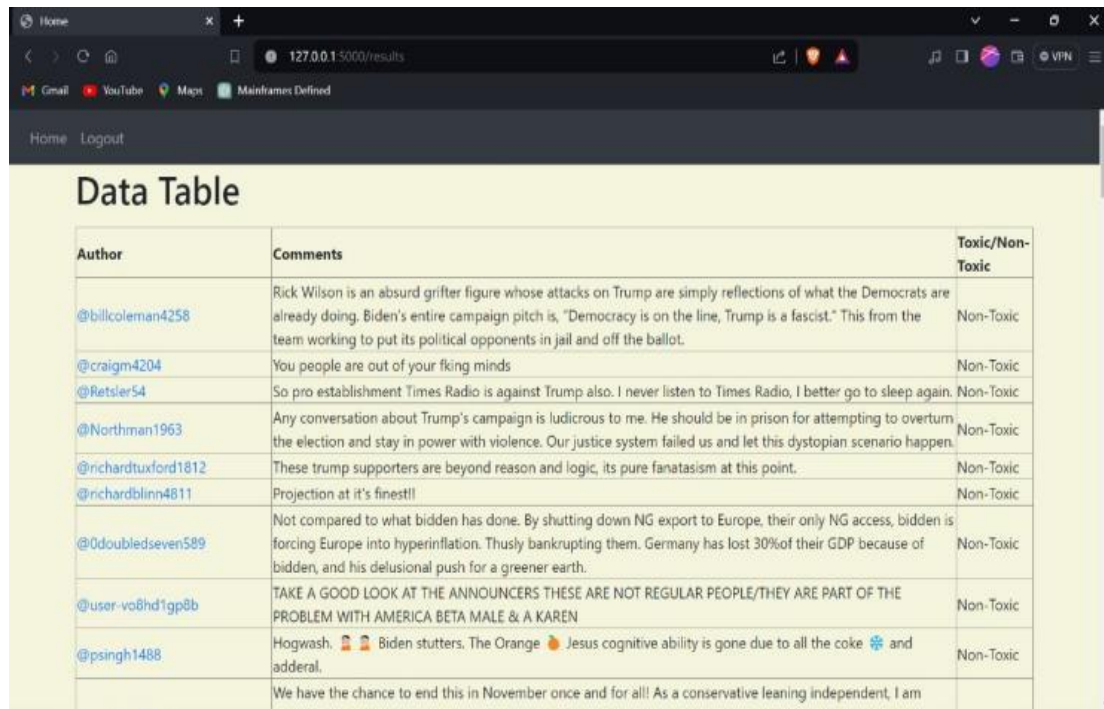
```
def link(youtube_link):
    video_id=extract_video_id(youtube_link)
    if video_id:
        request = youtube.commentThreads().list(
            part='snippet',
            videoId=video_id,
            textFormat='plainText',
            maxResults=10
        )
        # print(request)
        while request:
            response = request.execute()
            # print(response)
            for item in response['items']:
                comment.append(item['snippet']['topLevelComment']['snippet']['textDisplay'])
                authorname.append(item['snippet']['topLevelComment']['snippet']['authorDisplayName'])
            # print([item])
            # print()
            if len(comment)>=100:
                break
            request = youtube.commentThreads().list_next(request, response)
            print(request)
            # tfidf=pickle.load(open("tf_idf.pkl","rb"))
            tfidf_path = r"C:\Desktop\Major project\Flask-Web-App-Tutorial-main\Flask-Web-App-Tutorial-
            main\social_media\tf_idf.pkl"
            tfidf = pickle.load(open(tfidf_path, "rb"))
            toxicity_model_path=r"C:\Desktop\Major project\Flask-Web-App-Tutorial-main\Flask-Web-App-Tutorial-
            main\social_media\toxicity_model.pkl"
            nb_model=pickle.load(open(toxicity_model_path,"rb"))
            n=len(comment)
            for i in range(n):
                text_input=tfidf.transform([comment[i]]).toarray()
                prediction=nb_model.predict(text_input)
                if(prediction==1):
                    toxic.append("Toxic")
                else:
                    toxic.append("Non-Toxic")
            # Table={'Authors ' : authorname,
            # 'comments ' : comment,
            # 'Toxicity': toxic}
            table={}
            j=0
            for i in authorname:
                table[i]=[comment[j],toxic[j]]
            j+=1
            return table
        # print(link('https://www.youtube.com/watch?v=dam0GPOAvVI')['Toxicity'])
```



```
# print(link('https://www.youtube.com/watch?v=3mZHj1tv8iY'))
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from os import path
from flask_login import LoginManager
db = SQLAlchemy()
DB_NAME = "database.db"
def create_app():
    app = Flask(__name_)
    app.config['SECRET_KEY'] = 'hjshjhdjah kjshkjdhjs'
    app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:/// {DB_NAME}'
    db.init_app(app)
    from .views import views
    from .auth import auth
    app.register_blueprint(views, url_prefix='/')
    app.register_blueprint(auth, url_prefix='/')
    from .models import User, Note
    with app.app_context():
        db.create_all()
    login_manager = LoginManager()
    login_manager.login_view = 'auth.login'
    login_manager.init_app(app)
    @login_manager.user_loader
    def load_user(id):
        return User.query.get(int(id))
    return app
def create_database(app):
    if not path.exists('website/' + DB_NAME):
        db.create_all(app=app)
    print('Created Database!')
```







Author	Comments	Toxic/Non-Toxic
@billcoleman4258	Rick Wilson is an absurd grifter figure whose attacks on Trump are simply reflections of what the Democrats are already doing. Biden's entire campaign pitch is, "Democracy is on the line, Trump is a fascist." This from the team working to put its political opponents in jail and off the ballot.	Non-Toxic
@craigm4204	You people are out of your fking minds	Non-Toxic
@Retsler54	So pro establishment Times Radio is against Trump also. I never listen to Times Radio, I better go to sleep again.	Non-Toxic
@Northman1963	Any conversation about Trump's campaign is ludicrous to me. He should be in prison for attempting to overturn the election and stay in power with violence. Our justice system failed us and let this dystopian scenario happen.	Non-Toxic
@richardtuxford1812	These trump supporters are beyond reason and logic, its pure fanatism at this point.	Non-Toxic
@rchardblinn4811	Projection at it's finest!!	Non-Toxic
@0doubledseven589	Not compared to what bidden has done. By shutting down NG export to Europe, their only NG access, bidden is forcing Europe into hyperinflation. Thusly bankrupting them. Germany has lost 30% of their GDP because of bidden, and his delusional push for a greener earth.	Non-Toxic
@user-vo8hd1gp8b	TAKE A GOOD LOOK AT THE ANNOUNCERS THESE ARE NOT REGULAR PEOPLE/THEY ARE PART OF THE PROBLEM WITH AMERICA BETA MALE & A KAREN	Non-Toxic
@psingh1488	Hogwash. 🗑️ Biden stutters. The Orange 🍊 Jesus cognitive ability is gone due to all the coke 🧊 and adderal.	Non-Toxic
	We have the chance to end this in November once and for all! As a conservative leaning independent, I am	

8. TESTING

The testing of the "Toxic Comments Classification Using NLP" system involves various methodologies to ensure its reliability, accuracy, and effectiveness. Here are key testing methodologies employed during different stages of development:

TYPES OF TESTS

Unit Testing: Test individual modules and components in isolation to ensure they function as intended. Verify that each unit of code, such as algorithms, preprocessing functions, and user interface components, produces the expected output.

Integration Testing: Evaluate the interaction between different modules and components to ensure seamless integration. Verify that the system components work together as a cohesive unit, detecting and resolving any issues related to data flow and communication.

Functional Testing: Validate that the system meets the specified functional requirements. Test core functionalities such as data analysis, predictive modeling, user interface interactions, and cross-disciplinary adaptability.

User Interface (UI) Testing: Assess the usability and user-friendliness of the interface. Ensure that users can easily navigate through the system, input data, and interpret results.

Performance Testing: Evaluate the system's responsiveness, scalability, and resource usage under different conditions. Test the application's ability to handle large datasets and complex computations without compromising performance.

Regression Testing: Verify that new updates or modifications do not adversely impact existing functionalities. Re-run previously conducted tests to ensure that any changes have not introduced new errors or broken existing features.

Security Testing: Assess the system's resistance to unauthorized access, data breaches, and other security vulnerabilities. Implement measures to safeguard sensitive data processed by the application.

Cross-Browser and Cross-Platform Testing: Ensure compatibility with various web browsers and operating systems. Test the system's functionality on different platforms to guarantee a consistent user experience.

Adaptive Learning and Continuous Improvement Testing: Assess the system's ability to adapt and improve over time. Validate that the adaptive learning mechanisms are effectively enhancing the system's predictive capabilities based on new data and scenarios.

User Acceptance Testing (UAT): Involve end-users, researchers, and scientists in the testing process to gather feedback on the system's usability and effectiveness. Ensure that the system aligns with user expectations and fulfills their requirements.

9. CONCLUSION

This study analyzes the performance of various machine learning models to perform toxic comments classification and proposes an ensemble approach called Lstm-cnn. The influence of an imbalanced dataset and balanced dataset using random under-sampling and over-sampling on the performance of the models is analyzed through extensive experiments. Two feature extraction approaches including TF-IDF are used to get the feature vector for models' training. Results indicate that models perform poorly on the imbalanced dataset while the balanced dataset tends to increase the classification accuracy. Besides the machine learning classifiers like SVM, RF, GBM, and LR, the proposed RVVC and RNN deep learning models perform well with the balanced dataset. The performance with an

over-sampled dataset is better than the under-sampled dataset as the feature set is large when the data is over-sampled which elevates the performance of the models. Results suggest that balancing the data reduces the chances of models over-fitting which happens if the imbalanced dataset is used for training. Moreover, TF-IDF shows better classification accuracy for toxic comments. The proposed ensemble approach LSTM-CNN demonstrates its efficiency for toxic and non-toxic comments classification. The performance of LSTM-CNN is superior both with the imbalanced and balanced dataset, yet, it achieves the highest accuracy of 0.97 when used with TF-IDF features. The performance comparison with state-of-the-art approaches also indicates that LSTM-CNN shows better performance and proves good on small and large feature vectors. Despite the better performance of the proposed ensemble approach, its computational complexity is higher than the individual models which is an important topic for our future research. Similarly, dataset imbalance can overstate the results because data balancing using or random under-sampling approach may have a certain influence on the reported accuracy. Moreover, we intend to perform further experiments on multi-domain datasets and run experiments on more datasets for toxic comment classification.

10. REFERENCES

- [1] H. M. Saleem, K. P. Dillon, S. Benesch and D. Ruths, "A Web of Hate: Tackling Hateful Speech in Online Social Spaces", 2017, [online] Available: <http://arxiv.org/abs/1709.10159>.
- [2] M. Duggan, "Online harassment 2017", Pew Res., pp. 1-85, 2017.
- [3] M. A. Walker, P. Anand, J. E. F. Tree, R. Abbott and J. King, "A corpus for research on deliberation and debate", Proc. 8th Int. Conf. Lang. Resour. Eval. Lr. 2012, pp. 812-817, 2012.
- [4] J. Cheng, C. Danescu-Niculescu-Mizil and J. Leskovec, "Antisocial behavior in online discussion communities", Proc. 9th Int. Conf. Web Soc. Media ICWSM 2015, pp. 61-70, 2015.
- [5] B. Mathew et al., "Thou shalt not hate: Countering online hate speech", Proc. 13th Int. Conf. Web Soc. Media ICWSM 2019, no. August, pp. 369-380, 2019.
- [6] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad and Y. Chang, "Abusive language detection in online user content", 25th Int. World Wide Web Conf. WWW 2016, pp. 145-153, 2016.
- [7] E. K. Ikonomakis, S. Kotsiantis and V. Tampakas, "Text Classification Using Machine Learning Techniques", August 2005.
- [8] M. R. Murty, J. V. Murthy and P. Reddy, "Text Document Classification based on Least Square Support Vector Machines with Singular Value Decomposition", Int. J. Comput. Appl, vol. 27, no. 7, pp. 21-26, 2011.
- [9] E. Wulczyn, N. Thain and L. Dixon, "Ex machina: Personal attacks seen at scale", 26th Int. World Wide Web Conf. WWW 2017, pp. 1391-1399, 2017.
- [10] H. Hosseini, S. Kannan, B. Zhang and R. Poovendran, "Deceiving Google's Perspective API Built for Detecting Toxic Comments", 2017, [online] Available: <http://arxiv.org/abs/1702.08138>.