||Jai Sri Gurudev||

**BGSKH Education Trust**

# BGS COLLEGE OF ENGINEERING AND TECHNOLOGY

**MAHALAKSHMIPURAM, BENGALURU - 560086**

| | |
|---|---|
| Branch | : Artificial Intelligence & Machine Learning |
| Subject | : Natural Language Processing |
| Subject Code | : BAI601 |
| Academic Year | : 2024 - 25 |

# “Next word prediction model”

**By:**

Name : “Manusha P Y”

USN : “1MP22AI026”

**Faculty:**

Dr Madhura Gangaiah

Professor, Dept. of AIML

## Abstract

In the digital era, social media platforms such as Twitter have become significant channels for public expression and opinion sharing. Analyzing the sentiment behind tweets provides valuable insights into public perception of events, products, services, and individuals. This project focuses on Twitter sentiment analysis, aiming to classify tweets into sentiment categories such as positive, negative, or neutral The approach involves collecting and preprocessing Twitter data, which includes cleaning text, removing noise (e.g., URLs, mentions, hashtags), and applying natural language processing techniques. A machine learning model is then trained using features extracted through TF-IDF vectorization to understand and predict the sentiment of unseen tweets. The performance of the model is evaluated using standard metrics like accuracy and classification report. This sentiment analysis system can be applied in various domains such as market research, political analysis, and customer feedback systems, enabling organizations and individuals to make data-driven decisions based on public opinion.

## Introduction

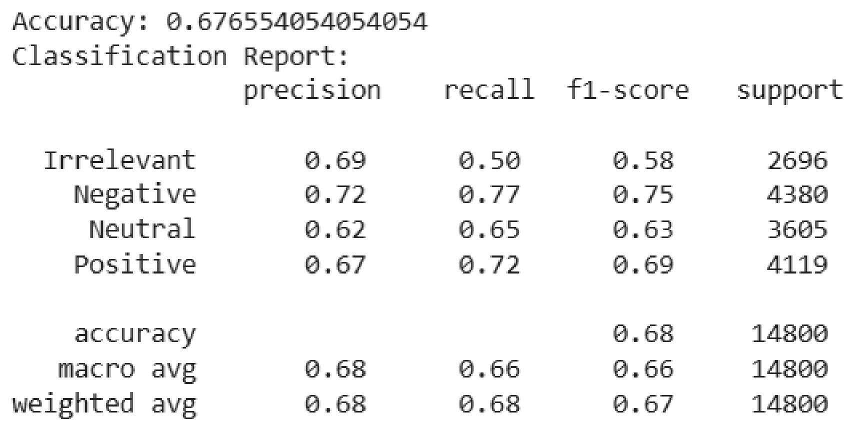
With the rapid growth of social media, platforms like Twitter have become essential for users to share opinions, reactions, and emotions on a wide range of topics, from current events to product reviews. These opinions, expressed in the form of short text messages (tweets), represent a rich source of real-time public sentiment. Sentiment analysis, also known as opinion mining, is a subfield of natural language processing (NLP) that focuses on identifying and categorizing emotions conveyed in text. By applying sentiment analysis to Twitter data, we can determine whether a tweet expresses a positive, negative, or neutral sentiment. This has numerous practical applications in areas such as business intelligence, brand monitoring, political forecasting, and customer service. This project aims to perform sentiment analysis on a labeled dataset of tweets. It involves several stages, including data cleaning, preprocessing, feature extraction using techniques like TF-IDF, and training a machine learning model to classify sentiment.

## Project Code

```
import pandas as pd

import re

import string

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, accuracy_score

from nltk.corpus import stopwords

import nltk

import os # Import the os module


# Download NLTK resources

# Download NLTK resources if not already present

try:

    stopwords.words('english')

except LookupError:

    nltk.download('stopwords')


# Check if the file exists and list directory contents

file_path = 'twitter_training.csv'


# Check if the file exists before trying to read it

if not os.path.exists(file_path):

    print(f"Error: File not found at {file_path}")

    print("Contents of /mnt/data/:")

    # Use a shell command to list directory contents to help the user

    !ls /mnt/data/
    # Exit the script or handle the missing file case appropriately
    # For now, we will print a message and the traceback will show the error,
    # but in a real application, you might want to sys.exit() or raise an exception.
else:
    print(f"File found: {file_path}")

    # Load dataset
    # This block is now inside the else statement, so it only runs if the file is found
    df = pd.read_csv(file_path, header=None, names=['tweet_id', 'entity', 'sentiment', 'text'])

    # Check for missing values
    df.dropna(inplace=True)

    # Preprocessing function
    def preprocess_text(text):
        # Ensure text is a string before processing
        if not isinstance(text, str):
            return "" # Return an empty string for non-string inputs

        text = text.lower()
        text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # remove URLs
        text = re.sub(r'\@w+|\#','', text)  # remove mentions and hashtags
        text = text.translate(str.maketrans('', '', string.punctuation))  # remove punctuation
        tokens = text.split()
        tokens = [word for word in tokens if word not in stopwords.words('english')]
        return ' '.join(tokens)

    # Apply preprocessing
```

```
df['clean_text'] = df['text'].apply(preprocess_text)

# Features and Labels
X = df['clean_text']
y = df['sentiment']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train model
model = LogisticRegression()
model.fit(X_train_vec, y_train)

# Predictions
y_pred = model.predict(X_test_vec)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

## Output



```
Accuracy: 0.676554054054054
Classification Report:
               precision    recall  f1-score   support

  Irrelevant       0.69      0.50      0.58      2696
    Negative       0.72      0.77      0.75      4380
     Neutral       0.62      0.65      0.63      3605
    Positive       0.67      0.72      0.69      4119

    accuracy                           0.68     14800
   macro avg       0.68      0.66      0.66     14800
weighted avg       0.68      0.68      0.67     14800
```