Hate Speech Detection

Abstract

This code implements a text classification pipeline to analyze tweets for hate speech, offensive language, and neutral content. It begins by importing necessary libraries, loading a dataset from a CSV file, and mapping numerical class labels to descriptive text labels. The dataset is then cleaned through a custom function that normalizes the text by converting it to lowercase, removing URLs, HTML tags, punctuation, digits, and stopwords, and applying stemming. The cleaned tweets and their corresponding labels are converted to NumPy arrays for further processing.

Using the CountVectorizer, the text data is vectorized into a format suitable for machine learning algorithms. The dataset is split into training and testing sets, and a Decision Tree Classifier is trained on the training data. The model's performance is evaluated using a confusion matrix, and accuracy is calculated. A heatmap visualizes the confusion matrix to illustrate the classification results. Finally, the code includes functionality for user input, allowing for real-time classification of tweets based on the trained model.

Objective

The objective of this code is to implement a text classification model to identify and categorize tweets based on their language content, specifically focusing on Hate Speech, Offensive Language, and Neutral Language. Initially, the code imports essential libraries and loads a dataset from a CSV file, ensuring data integrity by checking for missing values and providing structural insights.

Next, it enhances the dataset by mapping numerical class labels to descriptive categories, improving interpretability. The code then defines a data cleaning function that preprocesses tweets, transforming them to lowercase, removing URLs, HTML tags, punctuation, and numeric content, while also eliminating stop words and applying stemming to reduce words to their root forms.

The cleaned text is converted into a numerical format suitable for machine learning through vectorization, and the dataset is split into training and testing subsets. A Decision Tree Classifier is trained on the training data and evaluated on the test data, utilizing a confusion matrix and accuracy score to assess performance.

Finally, the model facilitates real-time predictions by allowing user input, demonstrating its practical application in monitoring online discourse. This project contributes to efforts in automatically identifying harmful language in social media.

Introduction

This document presents a comprehensive approach to analyzing and classifying tweets based on their linguistic content, specifically targeting hate speech and offensive language. The implementation leverages the Python programming language, utilizing powerful libraries such as Pandas, NumPy, and Scikit-learn, which are integral for data manipulation, analysis, and machine learning tasks.

The process begins with the loading of a dataset containing tweets labeled with corresponding classes: Hate Speech, Offensive Language, and No Hate/Offensive Language. To enhance interpretability, the numerical labels are mapped to descriptive text categories. The subsequent text preprocessing phase involves cleaning the tweets through lowercasing, removal of URLs, punctuation, and stop words, alongside applying stemming techniques to normalize the text.

Following preprocessing, the tweets are vectorized to convert textual data into numerical format, suitable for machine learning algorithms. A Decision Tree Classifier is employed to build a predictive model that identifies the class of each tweet. The model's performance is evaluated using a confusion matrix and accuracy metrics. This framework not only facilitates the identification of harmful content but also serves as a foundation for developing tools to monitor and mitigate online hate speech.

Methodology

The methodology employed in this code follows a systematic approach to classify tweets based on their language content.

1. **Data Collection and Preprocessing**: The dataset, sourced from a CSV file, contains tweets labeled with numerical classifications. Initial steps include loading the dataset, checking for missing values, and mapping numerical labels to descriptive text categories (Hate Speech, Offensive Language, and No Hate or Offensive Language).

2. **Text Cleaning**: A comprehensive text-cleaning function is implemented to preprocess the tweets. This includes converting text to lowercase, removing URLs, HTML tags, punctuation, numeric content, and stop words. Stemming is also applied to reduce words to their base forms, ensuring uniformity in the data.

3. **Feature Extraction**: The cleaned tweets are transformed into numerical vectors using the `CountVectorizer`, facilitating the application of machine learning algorithms.

4. **Data Splitting**: The dataset is divided into training and testing sets, with 80% allocated for training and 20% for testing. This allows for effective model evaluation.

5. **Model Training and Evaluation**: A Decision Tree Classifier is trained on the training data. Predictions are made on the test set, followed by the evaluation of the model's performance using a confusion matrix and accuracy score. Finally, the model is tested with user-inputted tweets to demonstrate its practical applicability.

Code

```
# Importing libraries
import pandas as pd
import numpy as np

# Load the dataset
dataset = pd.read_csv("Hate_data.csv")

#print(dataset.isnull().sum())
#print(dataset.info())
#print(dataset.describe())

# Add a 'labels' column that maps numerical class to meaningful text
labels
dataset["labels"] = dataset["class"].map({0: "Hate Speech",
                                          1: "Offensive Language",
                                          2: "No Hate or Offensive
Language"})

#print(dataset)

# Select only the 'tweet' and 'labels' column
data = dataset[["tweet","labels"]]

#print(data)

import re
import nltk
import string

#Importing of stop words
from nltk.corpus import stopwords
stopwords = set(stopwords.words("english"))

#Importing of stemming
stemmer = nltk.SnowballStemmer("english")
```

```python
# Define a function to clean the text data
def clean_data(text):
    text = str(text).lower()
    text = re.sub(r"http?://\S+|www\.\S+", "", text)
    text = re.sub(r"<.*?>+", "", text)
    text = re.sub(r"[%s]" % re.escape(string.punctuation), "", text)
    text = re.sub(r"\n", "", text)
    text = re.sub(r"\w*\d\w*", "", text)
    text = re.sub(r"rt", "", text)

    #Stopwords removal
    text = [word for word in text.split(" ") if word not in stopwords]
    text = " ".join(text)

    #Stemming the text
    text = [stemmer.stem(word) for word in text.split(" ")]
    text = " ".join(text)

    return text

# Apply the 'clean_data' function to clean the 'tweet' column
data.loc[:, "tweet"] = data["tweet"].apply(clean_data)

#print(data)

# Convert the 'tweet' and 'labels' columns to numpy arrays
x = np.array(data["tweet"])
y = np.array(data["labels"])
#print(x)
#print(y)

# Import libraries for vectorizing text and splitting data
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

cv = CountVectorizer()
x = cv.fit_transform(x)
#print(x)

# Split data into training and testing sets (80% training, 20% testing)
```

```python
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
#print(x_train)

#Building ML model
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)

y_pred = dt.predict(x_test)

#Confusion matrix and accuracy
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
#print(cm)

# Visualize the confusion matrix using a heatmap
import seaborn as sms
import matplotlib.pyplot as ply


sms.heatmap(cm, annot=True, fmt=".1f", cmap="YlGnBu")
#ply.show()

from sklearn.metrics import accuracy_score

# Calculate the accuracy of the model
accuracy_score(y_test,y_pred)
#print(accuracy_score(y_test,y_pred))

#sample = "Let's unite and kill all the people who are protesting
against the government"
#sample = clean_data(sample)
#print(sample)

#data1 = cv.transform([sample]).toarray()
#print(data1)

#dt.predict(data1)
#print(dt.predict(data1))
```

```
# Sample testing using user input
test_tweet = str(input("Input tweet to be sent : "))
test_tweet = clean_data(test_tweet)
test_data = cv.transform([test_tweet]).toarray()
dt.predict(test_data)
print(dt.predict(test_data))
```

Dataset used: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset

Conclusion

In conclusion, the implemented code effectively demonstrates the process of classifying tweets into predefined categories of language, specifically targeting hate speech and offensive language detection. By utilizing a structured approach that includes data preparation, text preprocessing, feature extraction, and machine learning model training, the code showcases how natural language processing (NLP) techniques can be applied to tackle real-world problems in online communication.

The data cleaning process enhances the quality of the input text by removing irrelevant elements, which is crucial for improving the accuracy of the model. The Decision Tree Classifier employed in this project allows for efficient training and testing, yielding a confusion matrix that provides insight into the model's performance. The achieved accuracy score reflects the model's ability to differentiate between the various classes of language, offering a reliable tool for identifying harmful content.

Furthermore, the inclusion of a user-input feature enables real-time classification, making the tool practical for monitoring social media platforms. Overall, this project not only highlights the capabilities of machine learning in natural language processing but also contributes to the ongoing efforts to promote safer online environments by automating the detection of hate speech and offensive language.