SATHYABAMA INSTITUTE OF SCIENCE & TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCSA 2604 NATURAL LANGUAGE PROCESSING LAB

LAB 5: SENTIMENT ANALYSIS

AIM: To perform sentiment analysis program using an SVM classifier with TF-IDF vectorization.

PROCEDURE:

Data Preparation: Downloading the dataset, converting it into a suitable format (words and sentiments), and structuring it into a DataFrame.

Splitting Data: Dividing the dataset into training and testing sets to train the model on a portion and evaluate it on another.

TF-IDF Vectorization: Converting text data into numerical vectors using TF-IDF (Term Frequency-Inverse Document Frequency) representation.

SVM Initialization and Training: Setting up an SVM classifier and training it using the TF-IDF vectors obtained from the training text data.

Prediction and Evaluation: Transforming test data into TF-IDF vectors, predicting sentiment labels, and evaluating the model's performance by comparing predicted labels with actual labels using accuracy and a classification report.

The following algorithm outlines the process of building a sentiment analysis model using an SVM classifier with TF-IDF vectorization in Python. Adjustments can be made to use different datasets, vectorization techniques, or machine learning models based on specific requirements.

ALGORITHM:

1. Library Installation and Import: Install required libraries (scikit-learn and nltk). Import necessary modules from these libraries.
2. Download NLTK Resources: Download the movie_reviews dataset from NLTK.
3. Load and Prepare Dataset: Load the movie_reviews dataset.
   Convert the dataset into a suitable format (list of words and corresponding sentiments) and create a DataFrame.
4. Split Data into Train and Test Sets: Split the dataset into training and testing sets (e.g., 80% training, 20% testing).

5. TF-IDF Vectorization: Initialize a TF-IDF vectorizer.
   Fit and transform the training text data to convert it into numerical TF-IDF vectors.
6. Initialize and Train SVM Classifier: Initialize an SVM classifier (using a linear kernel for this example).
   Train the SVM classifier using the TF-IDF vectors and corresponding sentiment labels.
7. Prediction and Evaluation: Transform the test text data into TF-IDF vectors using the trained vectorizer.
   Predict sentiment labels for the test data using the trained SVM classifier.
   Calculate the accuracy score to evaluate the model's performance.
   Generate a classification report showing precision, recall, and F1-score for each class.

PROGRAM:

```
# Install necessary libraries

!pip install scikit-learn

!pip install nltk


# Import required libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report

from nltk.corpus import movie_reviews  # Sample dataset from NLTK


# Download NLTK resources (run only once if not downloaded)

import nltk

nltk.download('movie_reviews')


# Load the movie_reviews dataset

documents = [(list(movie_reviews.words(fileid)), category)

        for category in movie_reviews.categories()

        for fileid in movie_reviews.fileids(category)]
```

```
# Convert data to DataFrame
df = pd.DataFrame(documents, columns=['text', 'sentiment'])

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['sentiment'], test_size=0.2,
random_state=42)

# Initialize TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train.apply(' '.join))

# Initialize SVM classifier
svm_classifier = SVC(kernel='linear')

# Train the classifier
svm_classifier.fit(X_train_tfidf, y_train)

# Transform the test data
X_test_tfidf = tfidf_vectorizer.transform(X_test.apply(' '.join))

# Predict on the test data
y_pred = svm_classifier.predict(X_test_tfidf)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display classification report
print(classification_report(y_test, y_pred))
```

OUTPUT:

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from
scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from
scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (3.2.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk)
(8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk)
(1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages
(from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk)
(4.66.1)
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
Accuracy: 0.84
```

	precision	recall	f1-score	support
neg	0.83	0.85	0.84	199
pos	0.85	0.82	0.84	201
accuracy			0.84	400
macro avg	0.84	0.84	0.84	400
weighted avg	0.84	0.84	0.84	400