SATHYABAMA INSTITUTE OF SCIENCE & TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCSA 2604 NATURAL LANGUAGE PROCESSING LAB

LAB 3: CASE STUDY

AIM: Customer Support Email Classification

Problem Statement:

A customer support company receives a large volume of incoming emails from customers with various inquiries, complaints, and feedback. Manually categorizing and prioritizing these emails is time-consuming and inefficient. The company wants to develop a text classification system to automatically classify incoming emails into predefined categories, allowing for faster response times and better customer service.

Objectives :

- The text classification system successfully categorizes incoming customer emails into predefined categories.
- It improves the efficiency of the customer support team by automating email classification and prioritization.
- The company can respond to customer inquiries and issues more promptly, leading to higher customer satisfaction and retention.

Dataset:

The company has a dataset of past customer emails along with their corresponding categories. Each email is labeled with one or more categories, indicating the type of inquiry or issue raised by the customer. For demonstration purposes, we will use the fetch_20newsgroups dataset from scikit-learn, which contains a collection of newsgroup documents, spanning 20 different newsgroups. We'll simulate this dataset as if it were customer support emails categorized into predefined categories.

Approach:

Data Preparation:

- Load the 20 Newsgroups dataset as a proxy for customer support emails.
- Select a subset of categories that represent different types of customer inquiries, complaints, and feedback.
- Prepare the data and target labels from the dataset.

Data Preprocessing:

- Clean the email text data by removing unnecessary information such as email headers, signatures, and HTML tags.
- Tokenize the text and convert it to lowercase.
- Remove stopwords and apply techniques like stemming or lemmatization to reduce words to their base forms.

Feature Extraction:

Use TF-IDF Vectorizer to convert text data into numerical features, limiting the maximum number of features to 10,000 and removing English stopwords.

Model Selection:

- Choose a suitable classification algorithm such as Linear Support Vector Classifier (LinearSVC) for text classification.
- Train the chosen model on the training data.

Model Evaluation:

- Predict labels for the test set using the trained model.
- Evaluate the classifier's performance using accuracy and a classification report, which includes precision, recall, and F1-score for each category.

Future Enhancements:

- Continuous monitoring and updating of the model to adapt to evolving customer inquiries and language patterns.
- Integration of sentiment analysis to assess the sentiment of customer emails and prioritize urgent or critical issues.
- Expansion of the model to handle multiclass classification and a wider range of customer inquiry categories.

Program :

```
# Import necessary libraries

from sklearn.datasets import fetch_20newsgroups

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.svm import LinearSVC

from sklearn.metrics import accuracy_score, classification_report
```

```
# Load the 20 Newsgroups dataset as a proxy for customer support emails
newsgroups = fetch_20newsgroups(subset='all', categories=['comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'rec.autos', 'rec.motorcycles', 'sci.electronics'])

# Prepare data and target labels
X = newsgroups.data
y = newsgroups.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create TF-IDF vectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_features=10000)
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Train the LinearSVC classifier
classifier = LinearSVC()
classifier.fit(X_train, y_train)

# Predict labels for the test set
predictions = classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, predictions, target_names=newsgroups.target_names))
```

Output:

```
Accuracy: 0.9389623601220752

Classification Report:
                          precision    recall  f1-score   support

comp.sys.ibm.pc.hardware       0.92      0.91      0.91       212
   comp.sys.mac.hardware       0.94      0.93      0.94       198
              rec.autos       0.97      0.93      0.95       179
        rec.motorcycles       0.96      0.99      0.97       205
        sci.electronics       0.92      0.93      0.92       189

               accuracy                           0.94       983
              macro avg       0.94      0.94      0.94       983
           weighted avg       0.94      0.94      0.94       983
```

Result:

This case study outlines the problem statement, dataset, approach, expected outcome, and future enhancements for developing a text classification system for customer support email classification. It demonstrates the application of machine learning techniques to automate and improve customer service processes.