EXPERIMENT NO 6

Aim: Implementation of Sentiment Analysis and Spam Filtering Techniques

Objective:- To understand and implement various sentiment analysis and spam filtering techniques for text data.

Description:

Sentiment Analysis:

Sentiment Analysis is a text analysis technique that enables companies to extract meaningful insights from qualitative data. By discerning positive and negative sentiment in text data such as tweets, product reviews, and support tickets, businesses can comprehend customer perceptions about their brand, product, or service. This comprehension leads to data-driven decisions and enhances customer experience.

Sentiment Analysis involves analyzing the emotions and perspectives conveyed by a speaker or author in a given piece of text. It utilizes language processing, linguistics, and text analytics to identify and extract subjective information from source materials. This technology delves into analyzing and predicting the latent information embedded within text, providing valuable insights into users' intentions, preferences, and sentiments.

Text can be categorized based on its subjective or objective nature. Sentences containing opinions or emotions are classified as subjective, while those conveying factual information are considered objective.

Examples:

1. **Subjective:** "This movie starring Tom Cruise and Angelina Jolie is great." (The sentence expresses an opinion about the movie, indicating subjective content.)
2. **Objective:** "This movie stars Tom Cruise and Angelina Jolie." (This sentence presents factual information without expressing any opinion or sentiment, making it objective.)

Subjective text can further be categorized into three broad classes based on the emotions conveyed:

- **Positive:** "I love watching Star series."
- **Negative:** "The movie was awful."
- **Neutral:** "I typically get hungry by evening." (While this sentence expresses a personal experience, it lacks any positive or negative sentiment and hence is considered neutral.)

Spam Filtering: S

pam filtering is a technique used to identify and segregate unsolicited or unwanted messages from legitimate ones. It is commonly employed in email systems, messaging platforms, and online forums to prevent users from being inundated with irrelevant or harmful content.

Spam filters employ various algorithms and techniques to analyze incoming messages and determine their likelihood of being spam. These techniques include keyword analysis, Bayesian filtering, machine learning algorithms, and blacklisting of known spam sources.

Examples of Spam:

1. "Congratulations! You've won a free cruise vacation! Click here to claim your prize."
2. "Get rich quick! Invest in our revolutionary money-making scheme today!"

Program:

Spam:

```
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

# Updated sample data

X_spam = [

   "Get a free iPad now!",

   "Limited time offer: buy one, get one free",

   "Claim your prize today",

   "Click here for more information",

   "Congratulations, you've won a trip to Hawaii!",

   "Exclusive discount for our loyal customers",

   "You've been selected for a special promotion",

   "Don't miss out on this amazing deal"

]

X_not_spam = [
    "Meeting postponed to next week",
    "Hey, how's it going?",
    "Check out our new collection",
    "Special discounts available",
    "Reminder: appointment tomorrow at 10am",
    "Thanks for your email, I'll get back to you soon",
    "Hope you're having a great day!",
    "See you at the party tonight"
]
X = X_spam + X_not_spam
y = [1] * len(X_spam) + [0] * len(X_not_spam)  # 1 for spam, 0 for not spam
# 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)
# Convert text data into numerical features using TF-IDF Vectorizer
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
# Train a Support Vector Machine classifier
classifier = SVC(kernel='linear')
classifier.fit(X_train_vec, y_train)
# Predict on the test set
y_pred = classifier.predict(X_test_vec)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)

print("Classification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
Accuracy: 0.25
Classification Report:
              precision    recall  f1-score   support

           0       0.25      1.00      0.40         1
           1       0.00      0.00      0.00         3

    accuracy                           0.25         4
   macro avg       0.12      0.50      0.20         4
weighted avg       0.06      0.25      0.10         4
```

Sentiment:

```
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

# Updated sample data

X_positive = [

    "I loved watching that movie, it was amazing!",

    "The food at the restaurant was delicious",

    "Today is a beautiful day",

    "This book is fantastic, I couldn't put it down",

    "I'm feeling really happy today",

    "Thank you for the wonderful experience!"

]

X_negative = [

    "Mondays are always so boring",
    "The customer service was terrible",

    "I'm feeling quite sad right now",

    "I didn't enjoy the movie, it was terrible",

    "The food tasted awful, I would not recommend it",

    "This has been a disappointing experience"

]

X = X_positive + X_negative

y = [1] * len(X_positive) + [0] * len(X_negative)  # 1 for positive sentiment, 0 for negative
sentiment

# 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)

# Convert text data into numerical features using TF-IDF Vectorizer

vectorizer = TfidfVectorizer()

X_train_vec = vectorizer.fit_transform(X_train)

X_test_vec = vectorizer.transform(X_test)

# Train a Support Vector Machine classifier

classifier = SVC(kernel='linear')

classifier.fit(X_train_vec, y_train)

# Predict on the test set

y_pred = classifier.predict(X_test_vec)

# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

print("Classification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
Accuracy: 0.3333333333333333
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.33      1.00      0.50         1

    accuracy                           0.33         3
   macro avg       0.17      0.50      0.25         3
weighted avg       0.11      0.33      0.17         3
```

Conclusion:

Techniques used for sentimental analysis and spam filtering are:

Techniques used for sentiment analysis include:

1. **Data Preprocessing:** This involves tasks such as tokenization, removing stop words, stemming or lemmatization, and handling of special characters and emojis. And for spam filtering, Gathering a corpus of spam and non-spam messages to train the spam filter.
2. **Feature Extraction:** Converting text data into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings (e.g., Word2Vec, GloVe). And for spam filtering, Techniques like TF-IDF Vectorizer, which represents the importance of each word in distinguishing between spam and non-spam messages.
3. **Model Selection:** Choosing an appropriate machine learning or deep learning model for sentiment classification. Common models include Support Vector Machines (SVM), Naive Bayes, Logistic Regression, and neural networks such as LSTM (Long Short-Term Memory) or CNN (Convolutional Neural Network).
4. **Model Training:** Training the selected model on labeled data (positive/negative sentiment) to learn patterns and associations between text features and sentiment labels. And for spam filtering, Training a Support Vector Machine (SVM) classifier on the TF-IDF transformed data to classify messages as spam or not spam.
5. **Evaluation:** Assessing the performance of the sentiment analysis model using metrics such as accuracy, precision, recall, and F1-score on a separate test dataset.. And for spam filtering, Assessing the performance of the spam filter using metrics such as accuracy, precision, recall, and F1-score on a separate test dataset.
6. **Fine-tuning:** Iteratively refining the model by adjusting hyperparameters, experimenting with different feature representations, or incorporating domain-specific knowledge to improve performance.