Assignment – 16

Gumma V L Prasad
(H.T.No: 2406CYS107)

1. You are tasked with developing a Python code for sentiment extraction utilizing a provided sample dataset. The dataset consists of textual data annotated with labels categorizing sentiments into four categories: "rude," "normal," "insult," and "sarcasm."
Dataset:
Real News: https://drive.google.com/file/d/1FL2HqgLDAP5550nd1h_8iBhAV-ISTnzr/view?usp=sharing
Fake News:
https://drive.google.com/file/d/1EdI_HyUeI_Fi2nld7rQnnGEpQqn_BwM-/view?usp=sharing

1. Outline the key steps involved in developing a sentiment extraction algorithm using Python.
2. Describe the structure and format of the sample dataset required for sentiment extraction.
3. Implement the Python code to read and preprocess the sample dataset for sentiment analysis. Ensure that the code correctly handles text data and labels.
4. Discuss the process of classifying sentiments into the specified categories: "rude," "normal," "insult," and "sarcasm." Explain any techniques or algorithms employed for this classification task.
5. Evaluate the effectiveness of the sentiment extraction algorithm on the provided sample dataset. Consider metrics such as accuracy, precision, recall, and F1-score.
6. Propose potential enhancements or modifications to improve the performance of the sentiment extraction algorithm. Justify your recommendations.
7. Reflect on the ethical considerations associated with sentiment analysis, particularly regarding privacy, bias, and potential misuse of extracted sentiments.
8. Write a complete code for this assignment.

Answer:-

Sentiment Analysis with Python

Here's a breakdown of the steps involved and the code for sentiment analysis:

1. Key Steps:

1. **Data Preprocessing:** Clean and prepare the text data by removing noise, punctuation, stop words, and converting text to lowercase.

2. **Feature Engineering:** Extract features from the text that represent sentiment. This can involve techniques like word embedding or bag-of-words.
3. **Model Selection & Training:** Choose a machine learning model like Naive Bayes, Support Vector Machines (SVM), or Recurrent Neural Networks (RNNs) and train it on the labeled dataset.
4. **Evaluation:** Assess the model's performance using metrics like accuracy, precision, recall, and F1-score.
5. **Prediction:** Use the trained model to predict sentiment labels for new unseen text data.

2. Sample Dataset Structure:

The dataset should consist of two columns:

- **Text:** This column contains the actual text data (sentences) for which sentiment needs to be determined.
- **Label:** This column contains the corresponding sentiment label for each text data point. The labels in this case are "rude," "normal," "insult," and "sarcasm."

3. Python Code for Reading and Preprocessing (text_preprocessing.py):

Python

```
import pandas as pd

def preprocess_data(data_path):
  """
  Reads the sentiment dataset, cleans text, and returns a DataFrame.

  Args:
     data_path (str): Path to the CSV file containing sentiment data.

  Returns:
     pd.DataFrame: DataFrame containing preprocessed text and labels.
  """
  data = pd.read_csv(data_path)

  # Clean text data (lowercase, remove punctuation, stop words)
  from nltk.corpus import stopwords
  from nltk.tokenize import word_tokenize
  stop_words = stopwords.words('english')

  def clean_text(text):
    text = text.lower()
    text = "".join([char for char in text if char.isalpha() or char == " "])
    words = word_tokenize(text)
    filtered_words = [word for word in words if word not in stop_words]
    return " ".join(filtered_words)
```

```
  data["Text"] = data["Text"].apply(clean_text)
  return data

# Example usage
data = preprocess_data("sentiment_data.csv")  # Replace with your data path
print(data.head())
```

4. Sentiment Classification Techniques:

Several algorithms can be used for sentiment classification:

- **Naive Bayes:** A simple and effective classifier that works well for text classification tasks.
- **Support Vector Machines (SVM):** More powerful than Naive Bayes, but requires careful parameter tuning.
- **Recurrent Neural Networks (RNNs):** Especially effective for capturing sentiment in longer text sequences due to their ability to learn long-term dependencies.

Choosing the right model depends on the dataset size, complexity, and desired accuracy.

5. Evaluation Metrics:

- **Accuracy:** Overall percentage of correctly classified sentiment labels.
- **Precision:** Ratio of true positives (correctly classified) to all positive predictions.
- **Recall:** Ratio of true positives to all actual positive cases in the data.
- **F1-Score:** Harmonic mean of precision and recall, combining both metrics.

These metrics are calculated using libraries like scikit-learn.

6. Performance Enhancements:

- **Feature Engineering:** Explore advanced techniques like n-grams (sequences of words) or TF-IDF (Term Frequency-Inverse Document Frequency) to capture more complex sentiment cues.
- **Model Selection & Tuning:** Experiment with different models (e.g., RNNs) and hyperparameter tuning to improve performance.
- **Data Augmentation:** Artificially increase the dataset size by generating variations of existing data points (e.g., synonyms, paraphrases) to improve model generalization.

7. Ethical Considerations:

- **Privacy:** Ensure user consent for data collection and anonymize data when possible.
- **Bias:** Sentiment analysis models can inherit biases from the training data. Mitigate bias by using diverse datasets and monitoring model performance across different categories.
- **Misuse:** Sentiment analysis can be misused for manipulation or profiling. Emphasize responsible use of extracted sentiment data.