TASK-3 REPORT

APPROACH-1(SVM)

Objective

The primary objective of this task is to classify text data into predefined categories using a Support Vector Machine (SVM) with a linear kernel. The implementation involves cleaning and preparing the data, training the model, and making predictions on a test dataset.

Steps Performed

1. Environment Setup

- Installed required libraries (`scikit-learn` and `pandas`) for data manipulation and model training.
- Mounted Google Drive to access datasets (`train3.csv` and `test3.csv`).

2. Dataset Loading

- Loaded the training dataset (`train3.csv`) from Google Drive.
- Verified successful loading with the shape of the dataset, which contained multiple rows and columns.

3. Data Cleaning

- **For the `category` column:**
    - Checked for missing (`NaN`) and infinite values.
    - Dropped rows with missing or invalid `category` values.
    - Converted the `category` column to integers after ensuring all values were valid.
- **For the `Text` column:**
    - Checked for missing (`NaN`) and empty string values.
    - Dropped rows with missing or empty `Text` values to ensure data quality.

    - Ensured the cleaned dataset was ready for processing, reducing rows accordingly.

4. Data Sampling

- Selected 20% of the cleaned dataset for computational efficiency, ensuring representativeness.
- Verified the sampled dataset size for both text and category columns.

5. Splitting Data

- Split the sampled dataset into training and testing sets:
    - **80% for training**
    - **20% for testing**
- Ensured no data leakage and maintained a consistent random state for reproducibility.

6. Feature Transformation

- Used `TfidfVectorizer` to convert text data into numerical features:
    - Extracted a maximum of 5000 features to limit dimensionality.
    - Fit the vectorizer on training data and transformed both training and testing datasets.

7. Model Training

- Trained an SVM model with a linear kernel on the TF-IDF-transformed training data.
- Enabled probabilistic outputs (`probability=True`) to support confidence scoring and additional use cases.

8. Model Evaluation

- Predicted categories on the test set using the trained model.
- Evaluated the model's performance using:
    - **Classification Report:** Included precision, recall, and F1-score for each category.
    - **Accuracy Score:** Measured the overall correctness of predictions.

9. Test Dataset Predictions

- Loaded the test dataset (`test3.csv`) from Google Drive.

- Cleaned the `Text` column in the test dataset by removing rows with missing or empty values.
- Transformed the cleaned test data using the previously trained `TfidfVectorizer`.
- Predicted categories for the test data using the trained SVM model.

10. Saving Results

- Saved the test predictions into a CSV file (`test3_predictions.csv`) on Google Drive, ensuring results are accessible for further analysis.

Results

- The SVM model achieved an accuracy of **85%**, demonstrating decent performance in sentiment classification tasks.

APPROACH-2(LLM)

Objective

The objective of this project is to perform sentiment analysis on a text dataset using a fine-tuned DistilBERT model. The process includes training, evaluation, and inference on unseen data.

Steps Followed

1. Data Loading and Preprocessing

- The dataset `train3.csv` was loaded from Google Drive using `pandas`.
- Textual data from the `Text` column and sentiment labels from the `category` column were used.
- Preprocessing steps included:
  - Dropping rows with missing values in `Text` or `category`.

- Converting `category` to numeric values and handling errors using `pd.to_numeric`.
- Ensuring `category` values start from 1 by incrementing each value by 1.
- Stripping extra spaces from the `Text` column.

- Split the dataset into training (5%) and testing (95%) subsets.

2. Data Splitting

- The training subset was further split into:
  - 80% training set.
  - 20% validation set.
- The `train_test_split` function was used for stratified splitting.

3. Model Selection

- The **DistilBERT** transformer model (`distilbert-base-uncased`) was chosen for classification.
- **Tokenizer**: Used `AutoTokenizer` to tokenize and preprocess text into token IDs and attention masks.
- **Model**: Used `AutoModelForSequenceClassification` with three output labels for multi-class classification.

4. Dataset Preparation

- Implemented custom PyTorch `Dataset` classes:
  - **`SentimentDataset`**: Handles training and validation datasets, including tokenization and label encoding.
  - **`SentimentDatasetTest`**: Handles test data for inference, focusing on tokenization without labels.
- Data was tokenized with:
  - `max_length=128`
  - `padding='max_length'`
  - `truncation=True`
- Used PyTorch's `DataLoader` to batch the data for efficient training and inference.

5. Model Training

- **Device**: The model was trained on GPU if available, otherwise on CPU.
- **Optimizer**: Used **AdamW** with a learning rate of `2e-5`.

- **Loss Function**: Used CrossEntropyLoss for multi-class classification.
- **Early Stopping**:
    - Monitored validation loss with a patience of 2 epochs.
    - Stopped training when validation loss stopped improving.
- **Epochs**: Trained for a maximum of 4 epochs, with progress tracked using `tqdm`.
- Training involved:
    - Forward pass for computing logits.
    - Backward pass for gradient computation.
    - Optimizer step for parameter updates.

6. Evaluation

- Evaluated the model on the validation set.
- Predicted labels were compared against true labels to compute accuracy using `accuracy_score`.

7. Saving Predictions

- Generated predictions for the test set and saved them to `test3_res.csv`.

8. Inference

- For inference on unseen data:
    - Loaded `test3.csv`.
    - Tokenized text data using the same tokenizer and `max_length=128`.
    - Used the trained model to predict sentiment labels.
    - Subtracted 1 from predicted labels to match the original label range.
    - Saved predictions to `test3_predictions.csv`.

Results

- The model achieved an **accuracy of 87.49%** on the validation dataset, indicating good performance in sentiment classification.
- Predictions for unseen data were saved in `test3_predictions.csv`, ready for further use.
- The workflow is effective and can be extended to handle larger datasets or similar tasks.