**Detailed Report on Model Training**

**1. Data Preprocessing:**

The dataset is loaded from a CSV file, which contains several columns, but only the text and target columns are relevant for sentiment analysis. The text field contains raw tweets, while target represents the sentiment label.

1. **Text Cleaning:**
   * Special characters, punctuation, and numbers are removed using regular expressions to focus on alphabetic characters.
   * This cleaning step is crucial as it eliminates irrelevant characters that don't contribute to the sentiment meaning.
2. **Tokenization:**
   * The tweet text is split into individual tokens (words). Tokenization helps break down text into meaningful parts that the model can process.
3. **Stopword Removal:**
   * Common stopwords like "and," "the," and "in" are removed to reduce noise in the text data. These words do not carry significant sentiment meaning and could affect the model's performance.
4. **Lemmatization:**
   * Each word is reduced to its base form using the WordNetLemmatizer. For example, words like "running" and "ran" are lemmatized to "run," ensuring uniformity and reducing vocabulary size.
   * This step is vital in normalizing the data and helps the model generalize better by reducing variations of the same word.

**2. Tokenization and Padding:**

The cleaned text is converted into sequences of integers using TensorFlow’s Tokenizer. This tokenizer assigns a unique integer to each word based on its frequency in the dataset. Words that don’t appear in the top 100,000 words are assigned a special out-of-vocabulary (OOV) token.

After tokenization, the sequences are padded to ensure uniform length:

* **Padding**: Tweets are padded to a fixed length (50 words). Shorter tweets are zero-padded at the end, while longer ones are truncated. Consistent sequence length is important since LSTMs require inputs of uniform shape.

**3. Splitting Data:**

The preprocessed data is split into training, validation, and test sets:

* **80% of the data** is used for training, ensuring the model learns from the majority of the dataset.
* **20% of the data** is set aside for testing, providing an unbiased evaluation of the model’s performance.
* An additional validation split (20% of the training set) is used to monitor model performance during training and tune hyperparameters, preventing overfitting.

**4. LSTM Model Architecture:**

The architecture is a sequential model with the following layers:

1. **Embedding Layer:**
   * Converts integer-encoded words into dense vectors of fixed size (100 in this case).
   * The embedding captures semantic information about words and their relationships, which is critical for understanding text.
2. **Two LSTM Layers:**
   * **First LSTM Layer**: Contains 128 units and returns sequences, allowing the next LSTM layer to process the entire sequence.
   * **Second LSTM Layer**: Contains 64 units and outputs a fixed-length vector. LSTMs are chosen because they are effective at capturing long-range dependencies in sequential data, making them ideal for sentiment analysis.
3. **Dense Layer:**
   * A fully connected layer with 64 units and a ReLU activation function. It introduces non-linearity to help the model learn complex patterns in the data.
4. **Dropout Layer:**
   * A dropout rate of 50% is applied to prevent overfitting by randomly dropping units during training. This forces the model to generalize better by preventing reliance on specific neurons.
5. **Output Layer:**
   * A single unit with a sigmoid activation function for binary classification (positive or negative sentiment). The sigmoid function outputs a probability between 0 and 1.

**5. Model Compilation:**

The model is compiled with:

* **Optimizer**: Adam optimizer, which is efficient and widely used for training deep learning models. It adapts the learning rate dynamically based on the gradients.
* **Loss Function**: Binary cross-entropy, which is suitable for binary classification tasks. It measures the difference between the predicted probability and the true label.
* **Metrics**: Accuracy is used to monitor performance during training and validation.

**6. Model Training:**

The model is trained for 5 epochs with a batch size of 64. The training process involves:

* **Backpropagation**: The model adjusts weights to minimize the binary cross-entropy loss.
* **Validation Data**: At each epoch, the model’s performance is evaluated on the validation set to monitor overfitting. If the validation loss increases while training loss decreases, it indicates overfitting, which can be controlled by regularization techniques like dropout.

**7. Saving the Model and Tokenizer:**

* The trained model is saved in a .keras file, allowing you to reload it for future predictions without retraining.
* The tokenizer, which maps words to integers, is saved using pickle. This ensures that the exact tokenization process can be used when applying the model to new data.

**8. Expected Results:**

During training, you would monitor:

* **Accuracy**: The proportion of correct predictions over the total predictions. With more epochs, the accuracy should increase until it converges.
* **Loss**: A lower loss indicates a better fit. Ideally, both training and validation loss decrease over time.

Upon completion, you would evaluate the model using:

* **Confusion Matrix**: To observe true positives, false positives, true negatives, and false negatives.
* **Classification Report**: To assess metrics like precision, recall, and F1-score, providing a detailed view of the model's performance on positive and negative sentiments.

**Final Thoughts:**

The LSTM-based model you have built is highly effective for sentiment analysis due to its ability to capture sequential patterns in text data. By using a combination of embeddings, LSTM layers, and dropout, the model is likely to generalize well to new data while avoiding overfitting. The use of a validation set ensures that the model’s performance is robust.

Top of Form

Bottom of Form