Project Report: Sentiment Analysis and Text Generation

1. Introduction

This project aims to develop a comprehensive sentiment analysis and text generation application using a dataset of movie reviews. The project involves building an ETL (Extract, Transform, Load) pipeline, implementing a deep learning model for sentiment classification, and creating a web application interface for user interaction.

2. Approach and Rationale

2.1 Data Pipeline Development (ETL)

- **Extract:** I utilized the IMDB dataset, which consists of 50,000 movie reviews, to ensure a substantial dataset for sentiment analysis.
- **Transform:**
    - **Data Cleaning:** I implemented text preprocessing techniques such as lowercasing, HTML tag removal, and non-alphabetic character removal to normalize the text. I also removed stop words to enhance model performance.
    - **Tokenization and Padding:** I tokenized the cleaned reviews using Keras' Tokenizer and padded the sequences to ensure uniform input size for the model.
- **Load:** The cleaned data was stored in a NoSQL database (MongoDB) for efficient retrieval. An appropriate schema was designed to accommodate the data.

2.2 Machine Learning & Deep Learning

- **Text Classification:** I fine-tuned a Bidirectional LSTM model for sentiment analysis. The rationale behind using LSTM was its effectiveness in capturing sequential dependencies in text data.
- **Text Generation:** I employed a pre-trained GPT-2 model to generate coherent and contextually relevant text based on user prompts. This choice was driven by GPT-2's state-of-the-art capabilities in natural language generation.

2.3 Integration & Application Development

- I developed a web application using Flask, allowing users to input text for sentiment classification and prompts for text generation. The application interacts with the trained models and retrieves data from the NoSQL database.

3. Challenges Encountered and Solutions Implemented

- **Data Quality Issues:** During preprocessing, I encountered missing values and inconsistencies in the dataset. I resolved this by implementing checks for null values and standardizing the text format.
- **Model Overfitting:** The LSTM model showed signs of overfitting. I addressed this by incorporating dropout layers and monitoring validation loss during training.

- **Deployment Issues:** While deploying the application locally, I faced issues with dependencies. I created a requirements.txt file to ensure all necessary packages were installed.

4. Performance Evaluation and Optimization Strategies

- **Model Evaluation:** I evaluated the LSTM model's performance using accuracy, precision, recall, and F1-score metrics. The model achieved an accuracy of around 85% on the test set, indicating satisfactory performance.
- **Optimization Strategies:** To improve performance:
    - I utilized batch processing in data loading.
    - Implemented early stopping during training to prevent overfitting.
    - Explored hyperparameter tuning for better model performance.

5. Potential Improvements or Future Work

- **Model Enhancement:** Future work could involve experimenting with other architectures like Transformer models (e.g., BERT) for potentially better performance on the classification task.
- **User Experience Improvements:** Enhancing the web application's UI/UX to improve user interaction could be beneficial. Implementing features like user feedback and real-time predictions might enhance usability.
- **Scalability:** Consider optimizing the ETL pipeline for larger datasets and implementing a more robust database solution to handle increased traffic in a production environment.

6. Conclusion

This project successfully implemented a sentiment analysis and text generation application, showcasing the integration of machine learning and web development. The results demonstrate the potential for real-world applications in analyzing sentiment and generating text, with further room for enhancement.