Sentiment Analysis Using Distributed Computing

Introduction and Objectives

This project aimed to create a complete data processing pipeline for analyzing large-scale datasets using distributed computing technologies. By leveraging Apache Spark, AWS Athena, and S3, we explored how big data tools can preprocess and analyze vast amounts of information efficiently. The specific focus of this project was sentiment analysis of Amazon product reviews, where the goal was to classify reviews into three sentiment categories: **positive**, **neutral**, or **negative**.

Data Source

The dataset used for this project was the **Amazon Product Reviews Dataset**, a collection of customer feedback on various products sold on Amazon. It includes over a million records with detailed review text and associated metadata. Here's why we chose this dataset:

- **Size:** At 287 MB and over 1M rows, it allowed us to test the scalability of distributed systems.
- **Diversity:** The dataset contains rich features like review text, helpfulness scores, and ratings, making it ideal for text analysis and feature engineering.
- **Real-World Relevance:** Understanding customer sentiment has practical applications for businesses in e-commerce.

Data Processing and Transformation

To make the dataset suitable for machine learning, we applied several preprocessing steps:

1. **Data Cleaning:**
    - We removed missing or irrelevant columns and filtered out rows with insufficient information.
2. **Feature Engineering:**
    - Created derived columns, including:
        - `word_count`: Total number of words in a review.

- `helpfulness_ratio`: Proportion of helpful votes to total votes.
- `avg_word_length`: Average length of words in a review.

- Tokenized the `Text` column to break it into individual words for analysis.
- Applied TF-IDF (Term Frequency-Inverse Document Frequency) to measure the importance of words.

3. **Data Storage:**

   - The preprocessed data was stored in AWS Athena for efficient querying and further processing. The output was saved in an S3 bucket: `s3://amzn-review-project/sent`

Machine Learning Model Development

Pipeline Overview

The machine learning workflow involved the following steps:

- Combined text-based features (`TF-IDF`) with numeric attributes like `word_count` and `helpfulness_ratio` into a single feature vector.
- Used logistic regression as the baseline model for classification.
- Split the dataset into training (80%) and testing (20%) sets.

Results and Insights

- The logistic regression model initially achieved an accuracy of **33%**, suggesting the need for further feature enrichment.
- By adding features like N-Grams and using Random Forest, accuracy improved to **60%**.
- Balancing the dataset helped reduce misclassification between neutral and other classes.

Evaluation and Observations

Key Findings

- Text-based features like `TF-IDF` and N-Grams were the most impactful for sentiment classification.
- Numeric features (`helpfulness_ratio`, `word_count`) contributed less to overall performance.
- Neutral sentiments were often misclassified as either positive or negative, highlighting the need for more granular text-based features.

Performance Metrics

- **Final Model Accuracy:** 60%.
- **Precision and Recall:** Improved for `positive` and `negative` classes after adding more text features.

Visualizations and Insights

Pipeline Overview

The pipeline integrates AWS and Spark components:

- Data flows from S3 to Spark for preprocessing.
- Preprocessed data is stored in Athena for further analysis.
- The final machine learning model is trained using Spark MLlib and results are stored back in S3.

Visualizations

We created several visualizations to better understand and interpret the model:

- A **Confusion Matrix** to visualize model performance across sentiment classes.
- A **Word Cloud** to highlight the most significant words for each sentiment.
- A **Feature Importance Chart** showing the impact of different features on predictions.

Challenges and Solutions

Challenges

- The dataset was slightly imbalanced, with fewer neutral reviews compared to positive and negative ones.
- Handling large TF-IDF matrices required significant memory and computational resources.
- Logistic regression as a baseline model had limited ability to capture complex relationships in text.

Solutions

- Applied undersampling to balance the sentiment classes.
- Used Spark's distributed computing capabilities to process large matrices efficiently.
- Improved feature representation with N-Grams and added polarity scores for better context.

Conclusion and Future Work

Conclusion

This project successfully demonstrated the integration of big data tools for processing and analyzing large datasets. By combining Spark and AWS services, we developed a scalable sentiment analysis pipeline. While the final model achieved a moderate accuracy of **60%**, it showcased the potential of distributed systems for real-world applications.

Future Work

1. Implement advanced NLP models like BERT or LSTMs for better understanding of review text.
2. Incorporate external data (e.g., product categories) for enhanced feature engineering.
3. Deploy the pipeline for real-time sentiment monitoring using AWS Lambda and API Gateway.

Attachments

- Complete PySpark scripts.
- Architecture diagrams.