MapReduce & Hive: Word-Based Sentiment Scoring and Trend Analysis On Historical Texts

By:

Team Cloudonauts

Introduction and Background

With the digitization of historical literature, it is now possible to apply computational methods to uncover patterns and insights from centuries-old texts. This project presents a sentiment analysis and trend detection pipeline built using Apache Hadoop MapReduce and Apache Hive. The goal is to analyze a collection of 18th and 19th-century books for word frequency, sentiment, and temporal trends, using distributed processing at scale. The workflow involves data cleaning, lemmatization, sentiment scoring with the AFINN lexicon, and the analysis of bigrams using a Hive UDF.

Objective

The primary objective of this project is to build a scalable data processing pipeline capable of extracting, analyzing, and interpreting sentiment and word trends from historical texts. The project is structured into five sequential tasks: text preprocessing, word frequency analysis, sentiment scoring, decade-wise trend aggregation, and bigram analysis using a custom Hive UDF.

Dataset and Tools

The dataset consists of five books from Project Gutenberg, each representing a different time period and literary genre. These books were stored in plain text and preprocessed to remove formatting and metadata. The sentiment lexicon used is AFINN-111, which provides integer sentiment values for common English words. The tools and frameworks used in this project include Java for MapReduce programming, Stanford CoreNLP for lemmatization, Apache Hive for data querying, and Hadoop for distributed processing. The Maven build system was used for project dependency management.

Task-wise Implementation

Task 1: Preprocessing MapReduce Job

The first task focuses on preparing the raw text for analysis by extracting useful metadata and cleaning the content. The Mapper extracts the book ID, title, and publication year, converts the content to lowercase, removes punctuation, and eliminates common stop words. The cleaned and tokenized text is grouped by (bookID, year) using the Reducer, which emits a structured and noise-free version of the dataset. The final output is written to 'output/Task1/part-r-00000', with a sample record such as: (pg1184, 1856) "sherlock holmes london mystery".

Task 2: Word Frequency Analysis with Lemmatization

This task builds on the preprocessed data and focuses on converting words into their base forms through lemmatization. Using Stanford CoreNLP, each sentence is tokenized, and every token is lemmatized to unify different word forms. The Mapper emits ((bookID, lemma, year), 1) for each word, and the Reducer aggregates these counts to determine the frequency of each lemma in the dataset. The results are stored in 'output/Task2/part-r-00000', for example: (pg1259, murder, 1890)

Task 3: Sentiment Scoring

In Task 3, sentiment analysis is performed using the AFINN-111 lexicon. The Mapper checks each lemmatized word against the lexicon and retrieves its sentiment score. If a word exists in the lexicon, the Mapper emits ((bookID, year), score). The Reducer sums all scores per (bookID, year) to provide an overall sentiment score for that text. This output is saved in 'output/Task3/part-r-00000', with entries like: (pg4085, 1815)

Task 4: Trend Analysis and Aggregation

This task introduces a temporal aspect to sentiment analysis by aggregating scores over decades. The Mapper converts each year to its corresponding decade (e.g., 1815 → 1810s), and the Reducer calculates the total or average sentiment for each decade. This enables the visualization of long-term trends in emotional tone across different time periods. The final output is saved in 'output/Task4/part-r-00000', such as: (1810s) Total Score: 89.

Task 5: Bigram Analysis Using Hive UDF

The final task involves extracting bigrams—pairs of consecutive words—from the lemmatized text. A custom Hive UDF written in Java is developed to accept text input, tokenize it, and generate bigrams. This UDF is registered and invoked within Hive queries on the output of Task 2. The analysis yields insights into common phrase structures used by authors.

Challenges Faced

Throughout the implementation of this project, various technical and architectural challenges were encountered. Each was addressed through debugging, configuration adjustments, and enhanced logic implementation. Below is a summary of key issues and their solutions:

Challenge	Solution
UDTF not allowed in WHERE/GROUP BY	Used LATERAL VIEW explode() properly with subqueries to handle UDTFs.
UDF compile errors (missing UDF class)	Added hive-exec dependency to pom.xml to ensure Hive could recognize the UDF.
Meaningless bigrams flooding results	Implemented strong stopword filtering for both words in each bigram.
0 results when using unique input	Removed HAVING frequency > 1 condition to allow analysis of rare bigrams.
Query only returning one book or decade	Used ROW_NUMBER() with PARTITION BY to extract top-N bigrams per group.
High memory usage during lemmatization	Reused Stanford CoreNLP objects and optimized JVM memory settings to reduce overhead.
Inconsistent formatting in input texts	Designed regex-based custom preprocessing logic to clean Project Gutenberg files.
Lexicon lookup inefficiency in MapReduce	Preloaded AFINN lexicon using Hadoop's Distributed Cache for fast access in all nodes.
Difficulty testing Hive UDFs locally	Created scaffolding for testing with sample Hive shell queries and local test data.

Results and Observations

The project successfully produced intermediate and final outputs that reveal sentiment trends, word usage patterns, and frequent bigrams across five historical books. Lemmas and sentiment scores were accurately computed per book and decade, showcasing the effectiveness of distributed processing for text analytics. The bigram analysis further illustrated repeated motifs and linguistic patterns in classic literature.

Conclusion

This project demonstrates the practical application of big data tools in performing large-scale textual analysis. By designing a structured, end-to-end processing pipeline with Hadoop MapReduce and Hive, the team extracted insights from complex historical datasets. The methodology can be extended to a broader corpus and enhanced with visualization tools or machine learning techniques to derive deeper understanding from literary texts.