Online Retail Product Recommendation System

Term Project Report

Hasan Emre Usta

202111301

Gökay Çetinakdoğan

202111050

Course: Big Data

Semester: Spring 2025

May 25, 2025

Contents

1 Abstract

This project implements a comprehensive big data analytics system for online retail data processing. The system utilizes Apache Spark for distributed data processing, Apache Kafka for real-time streaming, and MongoDB for data storage. The implementation focuses on building a scalable and efficient pipeline for processing large-scale retail data and generating personalized recommendations.

2 Introduction

2.1 Project Objectives

The primary objectives of this project are:

- To process and analyze large-scale retail transaction data
- To implement real-time data streaming capabilities
- To develop a machine learning-based recommendation system
- To create an interactive web interface for data visualization
- To ensure system scalability and performance

2.2 Project Scope

The project encompasses:

- Data ingestion and processing pipeline
- Real-time streaming implementation
- Recommendation system development
- Web application interface
- Performance optimization

3 System Architecture and Technologies

3.1 Apache Spark Implementation

Apache Spark was chosen as the primary data processing framework for several reasons:

- **Distributed Processing**

- Efficient handling of large-scale retail datasets (3 different CSV files)
  - In-memory computing capabilities for faster data processing
  - Parallel processing across multiple nodes for better performance
  - Optimized data partitioning for efficient data distribution

- **Key Spark Features Used**
  - Spark SQL for structured data processing of retail transactions
  - Spark Streaming for real-time data processing
  - MLlib for implementing ALS-based recommendation system
  - DataFrame API for efficient data manipulation and transformation

- **Implementation Details**
  - Data ingestion from multiple CSV files (online_retail.csv, online_retail_2009-2010.csv, online_retail_2010-2011.csv)
  - Efficient data union operations for combining datasets
  - Schema inference for automatic data type detection
  - Memory optimization through caching of frequently accessed data

3.2 Apache Kafka Integration

Kafka was implemented for real-time data streaming with the following considerations:

- **Purpose and Benefits**
  - Real-time data ingestion from multiple retail data sources
  - Reliable message queuing for data pipeline stability
  - Event streaming for continuous data processing
  - Fault tolerance and data consistency guarantees

- **Implementation Details**
  - Structured streaming pipeline from Spark to Kafka
  - JSON serialization for data transfer
  - Batch processing optimization for better throughput
  - Checkpoint mechanism for fault tolerance

- **Streaming Pipeline Architecture**

- Spark DataFrame to Kafka topic conversion
- Real-time data processing with Spark Structured Streaming
- Fault-tolerant data transfer to MongoDB
- Batch-based processing for better performance

3.3 MongoDB Database

MongoDB was selected for data storage with the following considerations:

- **Database Structure and Design**
  - Document-based storage for flexible data schema
  - Efficient storage of retail transactions
  - Optimized collections for recommendations
  - Indexed fields for fast query performance
- **Data Model Implementation**
  - Retail transactions collection for storing purchase data
  - User recommendations collection for personalized suggestions
  - Efficient storage of ALS model recommendations
  - Timestamp-based data organization
- **Performance Optimizations**
  - Batch operations for efficient data insertion
  - Index optimization for fast query execution
  - Connection pooling for better resource utilization
  - Efficient storage of recommendation results

4 Implementation Details

4.1 Data Processing Pipeline

The data processing pipeline consists of several key components:

4.1.1 Data Ingestion

- Loading multiple retail datasets
- Data validation and schema enforcement
- Initial data cleaning
- Data type conversion

4.1.2 Data Cleaning and Transformation

- Null value handling
- Date format standardization
- Negative value filtering
- Feature engineering
- Data normalization

4.2 Recommendation System

The recommendation system implements collaborative filtering using ALS:

- **ALS Model Implementation**
  - User-item interaction matrix construction
  - Matrix factorization for pattern discovery
  - Cold-start handling for new users/items
  - Efficient model training and prediction
- **Model Parameters and Optimization**
  - maxIter: 10 for balanced training time
  - regParam: 0.01 for regularization
  - alpha: 0.01 for implicit feedback
  - rank: 20 for feature representation
- **Performance Metrics**
  - RMSE evaluation for model accuracy
  - Training time optimization
  - Prediction speed for real-time recommendations
  - Scalability for large user bases

4.3 Web Application

The web interface was implemented using Flask:

- **Features**
  - Interactive recommendation display
  - Real-time data visualization
  - User-friendly interface
  - Efficient data retrieval from MongoDB
- **Implementation Details**
  - Flask-based REST API
  - MongoDB integration for data storage
  - Plotly for data visualization
  - Responsive web design

5 Performance Analysis

5.1 System Performance

- **Processing Performance**
  - Data ingestion speed
  - Processing throughput
  - Memory utilization
  - CPU usage
- **Recommendation System Performance**
  - Model training time
  - Recommendation generation speed
  - Prediction accuracy
  - System scalability

6 Results and Discussion

6.1 Key Achievements

- Successful implementation of distributed processing
- Efficient real-time data streaming
- Accurate recommendation generation
- Scalable system architecture

6.2 Challenges and Solutions

- **Technical Challenges**
  - Data volume handling
  - Real-time processing
  - System scalability
  - Performance optimization
- **Implemented Solutions**
  - Distributed processing with Spark
  - Kafka for real-time streaming
  - MongoDB for efficient storage
  - Optimized system configuration

7 Conclusion

7.1 Project Summary

The project successfully implements a comprehensive big data solution for retail analytics, demonstrating practical application of distributed computing, real-time processing, and machine learning. The system architecture and implementation details are summarized below:

7.1.1 System Architecture Achievements

- **Distributed Processing**
  - Successfully implemented Apache Spark for distributed data processing
  - Achieved efficient handling of large-scale retail datasets

  - Optimized memory usage and processing speed
  - Implemented effective data partitioning strategies
- **Real-time Processing**
  - Established reliable Kafka-based streaming pipeline
  - Achieved real-time data ingestion and processing
  - Implemented fault-tolerant message queuing
  - Maintained data consistency across the pipeline
- **Data Storage**
  - Implemented efficient MongoDB document storage
  - Optimized database queries and indexing
  - Achieved high data retrieval performance
  - Maintained data integrity and consistency

7.1.2 Technical Implementation Success

- **Data Processing Pipeline**
  - Successfully processed multiple retail datasets
  - Implemented comprehensive data cleaning procedures
  - Achieved efficient data transformation
  - Maintained data quality throughout the pipeline
- **Recommendation System**
  - Implemented ALS-based collaborative filtering
  - Achieved accurate personalized recommendations
  - Optimized model parameters for better performance
  - Successfully handled cold-start problems
- **Web Application**
  - Developed responsive and interactive interface
  - Implemented real-time data visualization
  - Achieved fast recommendation retrieval
  - Ensured smooth user experience

8 References

1. Apache Spark Documentation. (2023). https://spark.apache.org/docs/latest/
2. Apache Kafka Documentation. (2023). https://kafka.apache.org/documentation/
3. MongoDB Documentation. (2023). https://docs.mongodb.com/
4. Spark MLlib Documentation. (2023). https://spark.apache.org/docs/latest/ml-guide.html