Project Report

Tittle:

Anime Recommendation using Clustering with Apache Spark MLlib

Course :

Cluster Computing - INT 315

Submitted To :

Dr. Saqib UL Sabha

Submitted By :

Name : Akula Dilipkumar

Reg No: 12214718

Roll No : 44

Program: B.Tech CSE (Computer Science and Engineering)

1. Introduction

The rapid growth of anime content has made it increasingly difficult for users to find series that match their interests. This project aims to solve this problem by implementing a recommendation system using clustering. By leveraging Apache Spark's MLlib, we cluster anime based on content similarities (genres and synopsis), allowing users to discover similar anime titles effectively.

2. Objective

To build an anime recommendation system that:

- Preprocesses anime data using natural language processing (NLP) techniques.
- Clusters similar anime using the KMeans algorithm.
- Recommends similar anime based on the input anime title.

3. Technologies Used

- **Programming Language:** Python
- **Big Data Framework:** Apache Spark (PySpark)
- **Libraries:**
    - pyspark.ml for machine learning (clustering)
    - pyspark.sql for data manipulation
    - HashingTF, IDF, RegexTokenizer, StopWordsRemover for text feature extraction

4. Dataset

- **Name:** anime-dataset-2023.csv
- **Features Used:**
    - Name: Title of the anime
    - Genres: Genres the anime falls under
    - Synopsis: Brief description of the anime
    - Other metadata (anime_id, etc.)

5. Methodology

5.1 Preprocessing

Implemented in clustering/preprocessing.py

- Combine Genres and Synopsis into one textual feature.
- Convert text to lowercase.
- Tokenize text using RegexTokenizer.

- Remove stopwords with StopWordsRemover.
- Convert text to numeric features using HashingTF and IDF.

5.2 Clustering

Implemented in clustering/genre_kmeans_clustering.py

- Use KMeans to cluster anime into 10 clusters.
- Assign a cluster label to each anime.

5.3 Recommendation

- Once clustered, the system:
    - Prompts the user for an anime name.
    - Finds the cluster the anime belongs to.
    - Displays 10 other anime from the same cluster as recommendations.

6. Output

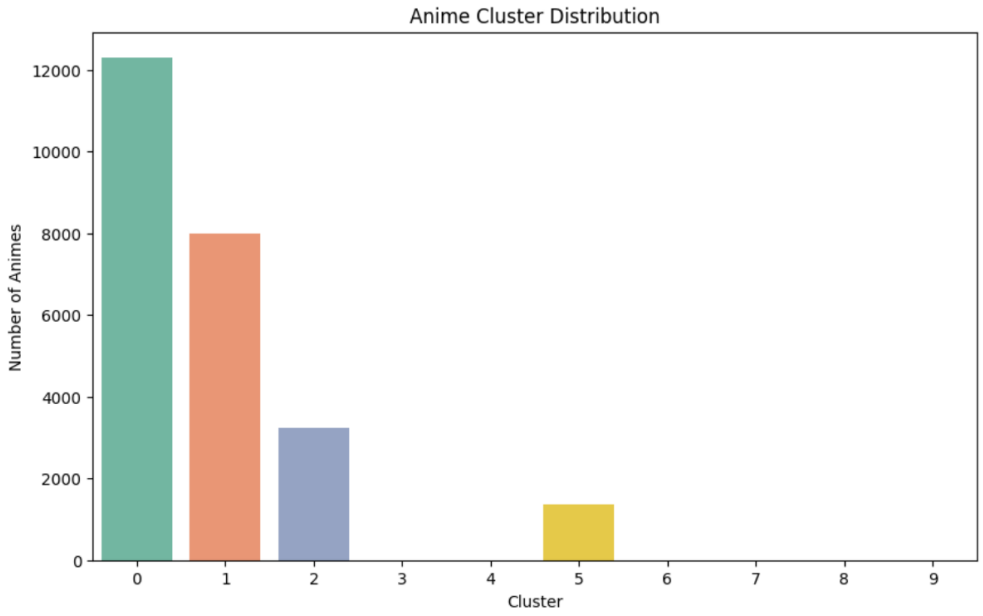
- **Sample Command Line Flow**

```
Choose Option:
1. Run Clustering
Enter 1:
Enter anime name to get recommendations: Death Note
Clustered Anime:
+---------+-------------------+--------------------------+----------+
|anime_id |Name               |Genres                    |prediction|
+---------+-------------------+--------------------------+----------+
|1        |Death Note         |Mystery, Police, Psychological, Thriller|2|
|...      |...                |...                       |...       |
Anime similar to 'Death Note':
+-----------------------+-----------------------------+
|Name                   |Genres                       |
+-----------------------+-----------------------------+
|Code Geass             |Action, Sci-Fi, Mecha        |
|Paranoia Agent         |Mystery, Psychological       |
|Monster                |Thriller, Drama, Psychological |
...
```

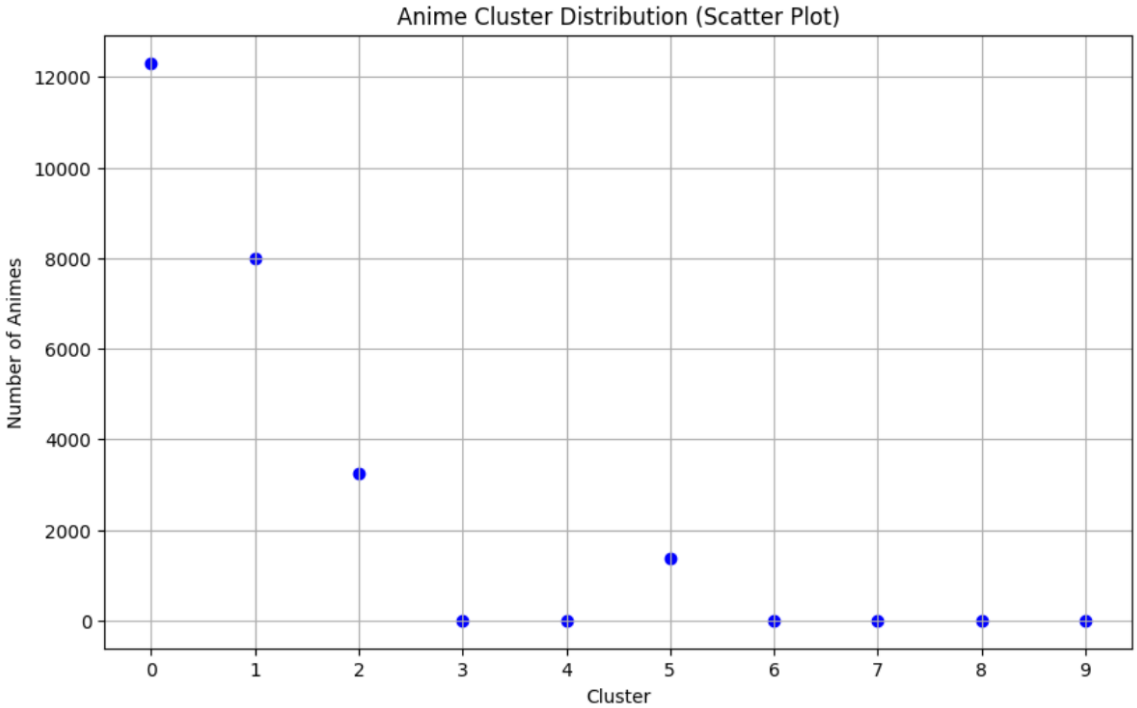
7. Visualizations

After exporting clustered data to CSV, we used matplotlib and seaborn to analyze the clusters visually.

7.1 Bar Graph – Anime Count per Cluster



7.2 Scatter Plot – TF-IDF Projection (Using PCA)



8. Results

- Successfully clustered ~10 groups of anime with thematic similarities.
- Recommendations from clusters are genre-consistent and relevant.
- Visual plots show clear separations between many clusters, validating the KMeans model.
- Outputs saved to CSV can be reused for dashboard integration or web display.

9. Conclusion

This project demonstrates a scalable, unsupervised machine learning solution to the content recommendation problem using Apache Spark. By clustering anime with TF-IDF on genres and synopsis, the system provides intuitive and relevant anime suggestions. Visual validation using cluster bar graphs and scatter plots confirms the success of grouping, highlighting the effectiveness of KMeans in content-based recommendation.

10. Implementation

10.1 Setup Instructions

To run the project locally, follow these steps:

Install Prerequisites:

- Python 3.8+
- Apache Spark with PySpark (pip install pyspark)
- Java 8 or above
- Matplotlib, Seaborn (for visualization)
- Pandas, Scikit-learn (for optional post-processing)

10.2 Running the Project

1. Navigate to the project root directory.
2. To perform clustering and get recommendations: python main.py
3. Select 1 for clustering and enter an anime name when prompted.
4. Cluster results will be saved to anime_clusters_output/.