Content Based Anime Recommender System Using PySpark and Kafka

Le Quoc Anh, Nguyen Doan Nhat Khanh, Do Trong Hop
Infomation of Technology University, ĐHQGHCM
{21522207, 21520565}@gm.uit.edu.vn
hopdt@uit.edu.vn

Abstract: The rapid growth of the anime industry, fueled by the increasing popularity of streaming platforms and diverse user preferences, has created a demand for personalized content recommendation systems. In this report, the research team proposed a Content-Based Anime Recommender System utilizing PySpark and Kafka to process and analyze large-scale user and anime data efficiently. The system leverages features such as genre, rating, and user preferences to recommend anime tailored to individual users. By integrating Kafka for real-time data streaming and PySpark for distributed data processing, the system ensures scalability and responsiveness. The evaluation of the recommender system was conducted using metrics such as Precision, Recall, and F1-Score, demonstrating its effectiveness in delivering accurate recommendations. The results indicate that the proposed system can significantly enhance user satisfaction by providing relevant anime recommendations in real-time. This approach highlights the potential of combining PySpark and Kafka in building scalable and efficient recommendation systems for the entertainment industry.
Keywords: Anime recommender system, Content-based filtering, PySpark, Kafka, Real-time recommendation.

1 Introduction

With the rapid growth of technology and online streaming platforms, the anime industry has expanded significantly, presenting challenges in delivering personalized content to users. The diverse preferences of viewers and the increasing variety of anime titles necessitate efficient recommendation systems. Content-based recommender systems use attributes such as genres, ratings, and user preferences to generate tailored recommendations. However, the scalability and efficiency of traditional methods are limited by the ever-growing volume of data.

This study proposes a Content-Based Anime Recommender System using PySpark and Kafka to address these challenges. PySpark enables efficient processing of large datasets, while Kafka facilitates real-time data streaming, ensuring scalability and responsiveness. The system leverages features like genres

and ratings to train models that deliver accurate, personalized recommendations. Performance is evaluated using metrics such as Precision, Recall, and F1-Score.

By integrating distributed data processing and real-time streaming technologies, this research provides a scalable solution to meet the demand for personalized anime recommendations, enhancing user experience in the digital age.

The structure of the paper is presented as follows:

- Section 2: Related Works. We present the relevant research works related to the topic of "Content Based Anime Recommender System Using PySpark and Kafka."
- Section 3: Methods. We discuss the dataset used by our team, the data preprocessing techniques employed, the models selected by our team, the methods for evaluating the models, and provide a detailed explanation of the processing system proposed by our team.
- Section 4: Results and Evaluation. We present the comparison results among the models, which serve as the basis for our team to select the most suitable model for streaming. Additionally, we provide an overview of the deployment of the model demo using Kafka streaming..
- Section 5: Conclusion. We present the conclusions drawn from the implementation of the aforementioned sections and provide insights into future directions for development.

2 Related Works

Understanding the importance of personalized recommendation systems in the anime industry is crucial due to the growing diversity of content and user preferences. Recommendation systems, particularly content-based methods, have gained significant attention in research for their ability to provide tailored suggestions. However, building efficient and scalable systems remains a challenge due to the vast and dynamic nature of user interactions and content metadata. The following are some notable research works in the field of recommendation systems:

- In the research paper *"Content-Based Anime Recommendation System Using Cosine Similarity"*, the authors explored the use of content-based filtering by analyzing anime metadata such as genres and synopses. Cosine similarity was employed to calculate the relevance between user preferences and available content, highlighting the effectiveness of feature-based recommendations.
- The research paper *"A Real-Time Recommendation System Using PySpark and Kafka"* demonstrated the integration of distributed data processing

frameworks and streaming technologies to enhance scalability and responsiveness. The study successfully utilized PySpark and Kafka to build a recommendation system for music streaming, which can be adapted for anime recommendations to process large datasets in real time.

- In *"Hybrid Recommendation Systems with Gated Recurrent Neural Networks"*, a hybrid approach combining collaborative filtering with content-based filtering was introduced. By employing Gated Recurrent Neural Networks (GRNN) alongside feature analysis, the model effectively addressed sparsity and cold-start issues, achieving higher accuracy in recommendations.

- The research paper *"Enhancing Content-Based Recommendations with Deep Learning"* applied advanced neural network architectures such as autoencoders and attention mechanisms to improve recommendation accuracy. These models captured complex relationships between users and items, resulting in more personalized and diverse suggestions.

- In *"Scaling Recommendation Systems Using Distributed Frameworks"*, the authors demonstrated the potential of distributed processing platforms such as PySpark in handling massive datasets efficiently. This research underscores the importance of scalable solutions for real-time recommendation systems in industries like anime streaming.

These studies demonstrate the effectiveness of various machine learning techniques, hybrid models, and distributed frameworks in building recommendation systems. This research builds upon these advancements by developing a Content-Based Anime Recommender System using PySpark and Kafka, addressing scalability and personalization challenges while enhancing user experience in the anime industry.

3 Methods

3.1 Dataset

The data is collected from AnimeVietsub.page. It is a platform dedicated to streaming anime content. The dataset in question was collected from this site and contains information about various anime series. It consists of twelve columns with the following details:

- Title: The name of the anime series or movie.
- Link: The URL directing to the anime's detail or streaming page.
- Image: The URL or file path to the cover or thumbnail image of the anime.
- Episode: The episode number or range of episodes available for the series.
- Rating: The user-generated score or rating for the anime.

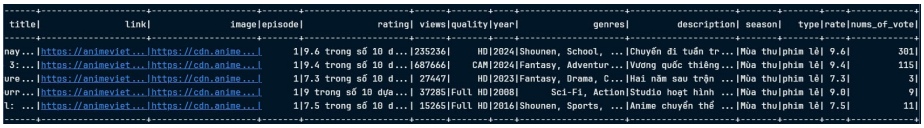
- Views: The number of views the anime has received on the platform.
- Quality: The video quality of the anime (e.g., 720p, 1080p).
- Year: The year the anime was released or aired.
- Genres: The list of genres associated with the anime (e.g., Action, Romance).
- Description: A brief summary or synopsis of the anime's plot.
- Season: The season the anime belongs to (e.g., Winter 2025, Spring 2024).
- Type: The category of the anime, such as TV series, OVA, Movie, or Special.

After the data cleaning process, the "Rating" column was replace by "id" and two more columns were generated, respectively:

- Rate: The average rating of the anime, calculated based on user reviews or feedback.
- Number of Votes: The total number of user votes or ratings submitted for the anime.

3.1.1 Anime Dataset

The dataset consists of 5380 rows and 14 columns with a period.



title	link	image	episode	rating	views	quality	year	genres	description	season	type	rate	nums_of_vote
nay...	https://animeviet...	https://cdn.anime...	1	9.6 trong số 10 d...	235236	HD	2024	Shounen, School, ...	Chuyến đi tuần tr...	Mùa thu	phim lẻ	9.6	301
3:...	https://animeviet...	https://cdn.anime...	1	9.4 trong số 10 d...	687666	CAM	2024	Fantasy, Adventur...	Vương quốc thiêng...	Mùa thu	phim lẻ	9.4	115
ure...	https://animeviet...	https://cdn.anime...	1	7.3 trong số 10 d...	27447	HD	2023	Fantasy, Drama, C...	Hai năm sau trận ...	Mùa thu	phim lẻ	7.3	3
urr...	https://animeviet...	https://cdn.anime...	1	9 trong số 10 dựa...	37285	Full HD	2008	Sci-Fi, Action	Studio hoạt hình ...	Mùa thu	phim lẻ	9.0	9
l: ...	https://animeviet...	https://cdn.anime...	1	7.5 trong số 10 d...	15265	Full HD	2016	Shounen, Sports, ...	Anime chuyển thể ...	Mùa thu	phim lẻ	7.5	11

Hình 1: Sample data

3.2 Data Processing

3.3 Model and Algorithm

3.3.1 Model: Content Page Filtering

Content Page Filtering is a machine learning model used for recommendation and filtering tasks. It focuses on analyzing the content of items, such as text, metadata, or features, to identify similarities and relevance to the user's preferences. The model aims to recommend items that closely match the user's interests based on their past behavior or explicitly stated preferences. Content Page Filtering is particularly useful in scenarios where user-item interactions are sparse, and the primary focus is on the features of the items rather than collaborative user data.

While there isn't a single formula for content filtering, it typically involves the use of algorithms to match content with specific criteria. One commonly used technique for measuring the relevance of content is **Cosine Similarity**, which calculates the similarity between two vectors (e.g., two content pages).

The formula for **Cosine Similarity** between two vectors A and B is as follows:

$$CosineSimilarity(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where:

- A and B are the vectors representing the content (e.g., term frequency vectors of two documents),
- $A \cdot B$ is the dot product of the two vectors,
- $\|A\|$ and $\|B\|$ represent the Euclidean norms (magnitudes) of the vectors.

Cosine Similarity measures the angle between two vectors. The result ranges from 0 (completely dissimilar) to 1 (completely similar), indicating the degree of content relevance.

3.3.2 Algorithm: Count Vectorizer

Count Vectorizer is a natural language processing algorithm used to convert text data into numerical features. It works by tokenizing the text into individual words (or n-grams) and counting the frequency of each word in the dataset. The result is a sparse matrix where each row represents a document and each column corresponds to a unique word in the vocabulary.
Count Vectorizer is particularly useful for preprocessing textual data for machine learning tasks, such as text classification or sentiment analysis, as it transforms raw text into a structured format suitable for modeling. The **Count Vectorizer** is a technique used to convert text data into numerical form, specifically creating a term frequency (TF) matrix. The **Term Frequency (TF)** of a word w in a document d is given by:

$$TF(w, d) = \frac{Numberoftimeswordwappearsindocumentd}{Totalnumberofwordsindocumentd}$$

Once the Count Vectorizer has created a term frequency matrix, it is often followed by applying **TF-IDF (Term Frequency-Inverse Document Frequency)** weighting to adjust the importance of words.

The formula for **TF-IDF** is:

$$TF{-}IDF(w, d) = TF(w, d) \times \log\left(\frac{N}{df(w)}\right)$$

Where:

- N is the total number of documents in the corpus,
- $df(w)$ is the number of documents containing the word w.

The purpose of the TF-IDF weighting is to reduce the influence of common words (e.g., "the", "and") and emphasize the significance of less frequent, yet more meaningful, terms.

3.4 Evaluation Methods

Precision@K

Precision@K is a widely used evaluation metric in information retrieval and machine learning, especially in ranking tasks. It measures the proportion of relevant items among the top K retrieved items. Precision@K focuses on assessing the quality of results at a specific cutoff rank, making it suitable for scenarios where only the top results matter, such as search engines, recommendation systems, and classification tasks.// The formula for **Precision@K** is given as:

$$Precision@K = \frac{NumberofRelevantItemsinTopK}{K}$$

Where:

- **Relevant Items**: The items that meet the criteria for relevance based on the ground truth or user preferences.
- **K**: The number of retrieved items considered in the evaluation.

The desired result for **Precision@K** is a value between 0 and 1, where:

- 1.0: Indicates perfect precision, meaning all K retrieved items are relevant.
- 0.0: Indicates no relevant items among the top K results.

Key Points:

- A high **Precision@K** indicates that the system is effective in presenting relevant results at the top.
- It focuses on *accuracy at the head of the ranking*, ignoring results beyond K.
- **Precision@K** is particularly useful in contexts like search engines, recommender systems, or ranked classifications, where only the top results matter to the user.

3.5 Processing system

Apache Kafka Streaming is a scalable and fault-tolerant system for processing real-time data. It is capable of handling high throughput and processing data while maintaining fault tolerance. Spark Streaming, on the other hand, is an extension of the Spark API that allows users to process real-time data from various sources such as Kafka, Flume, Kinesis, HDFS, TCP Sockets, and Twitter. It enables the processing of streaming data using complex algorithms expressed through high-level functions like reduce, join, and window.

Hình 2: Description of the Spark Streaming system

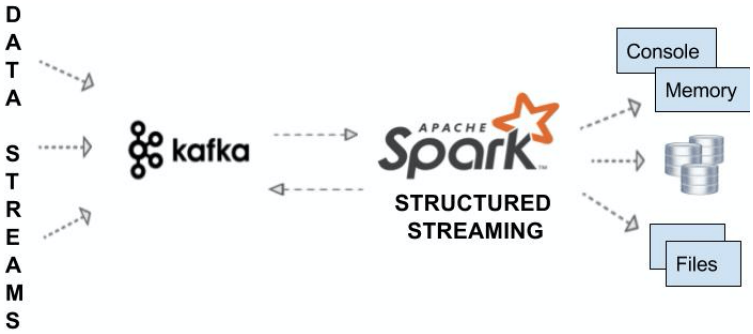
In terms of its operation, Spark Streaming directly receives input data streams and divides the data into batches, which are then processed by the Spark engine to generate streaming results based on these batches.

Hình 3: Description of how Spark Streaming works

Spark has the following four important characteristics:

- Quick recovery from failures and errors.
- Load balancing and efficient resource utilization.
- Integration of streaming data with static datasets and interactive queries.
- Native integration with advanced processing libraries (SQL, machine learning, graph processing).

The diagram illustrates a data processing system. Firstly, data is fetched from the exchange and recorded in a CSV file. It is then sent to a Kafka server for storage. Next, Apache Spark is used to retrieve data from the Kafka server and



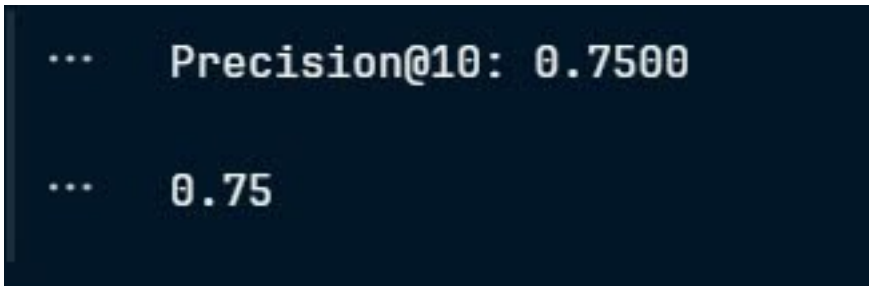
Hình 4: Description of the data processing process.

perform streaming processing. The ultimate result is the prediction of the closing price of the exchange for that day, outputted in memory. This output is combined with pre-existing similar data to display as a DataFrame or in a graphical representation, showing the changes in stock prices. This helps evaluate whether there will be future price growth in order to make informed observations.

4 Results and Evaluation

4.1 Training results of the models

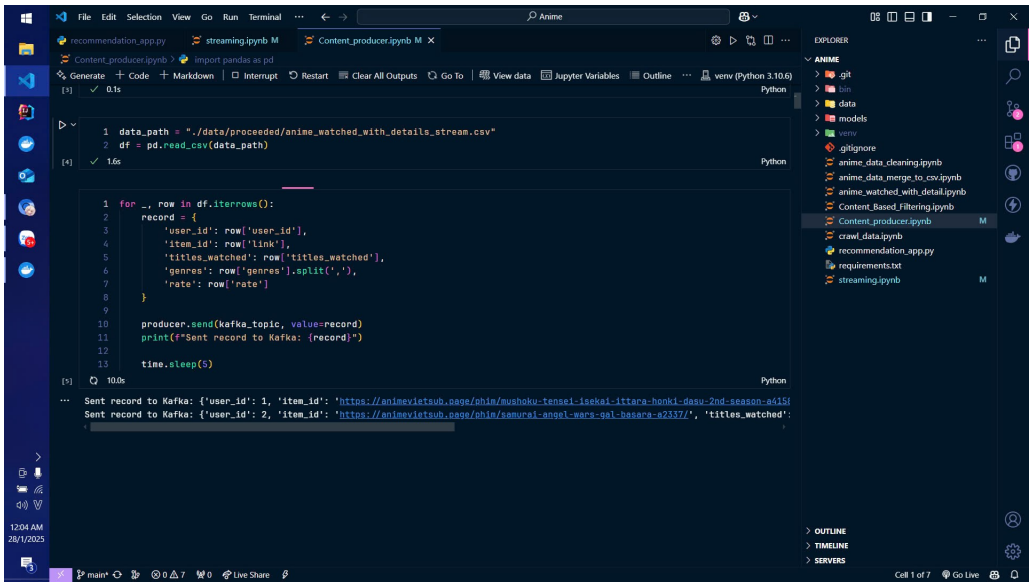
Below is the result of model evaluation using Precision@K:



Hình 5: Result of model evaluation using Precision@K

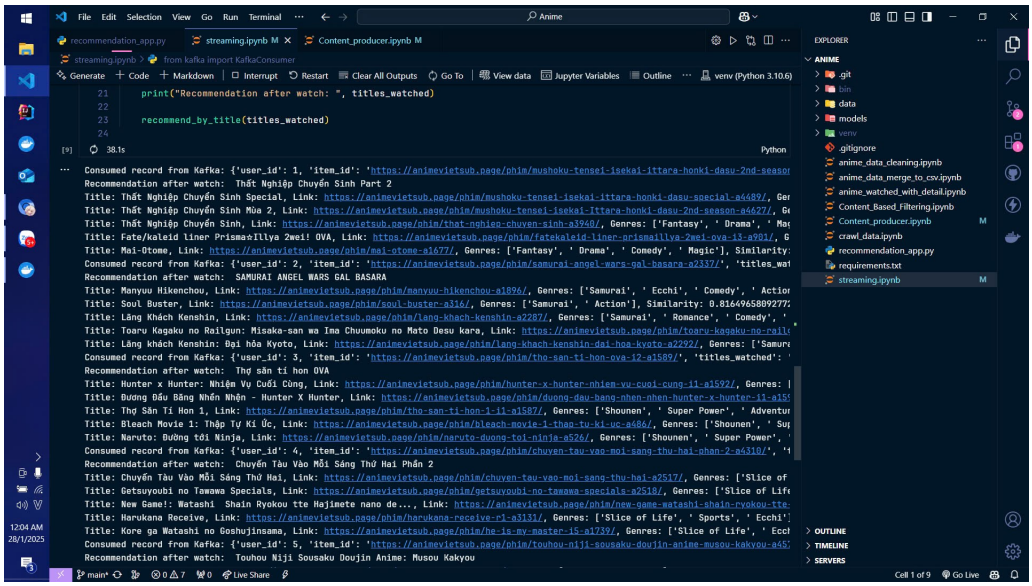
4.2 Deployment of the model demo

First, the producer sends "messages" to Kafka:



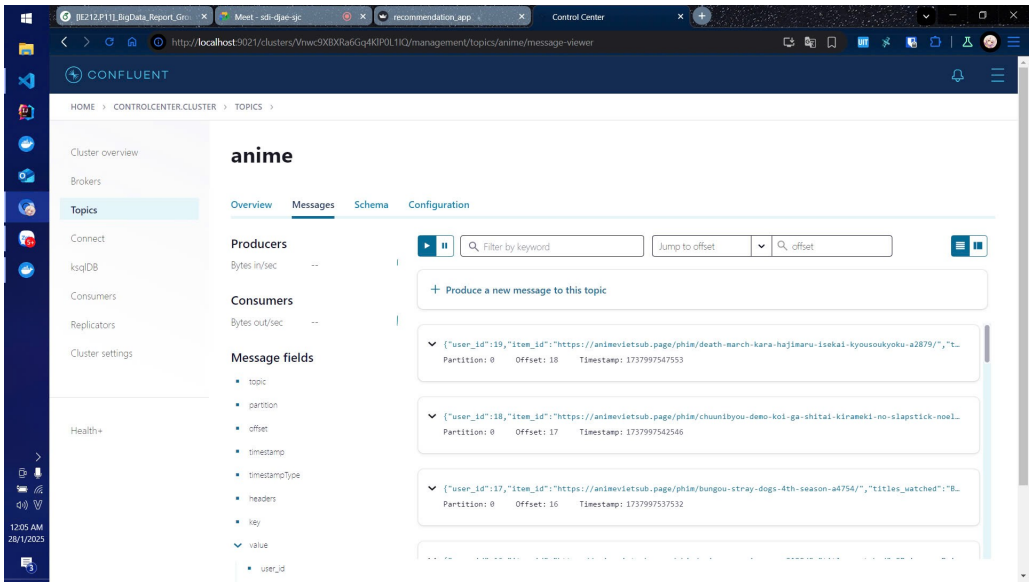
Hình 6: Producer

Then, the consumer gets data sent from Kafka. This data will be filtered to extract the titles of the movie the user just watched. After that, these "titles" will be fed into the trained model and an output containing recommended informations will be extracted:



Hình 7: Consumer

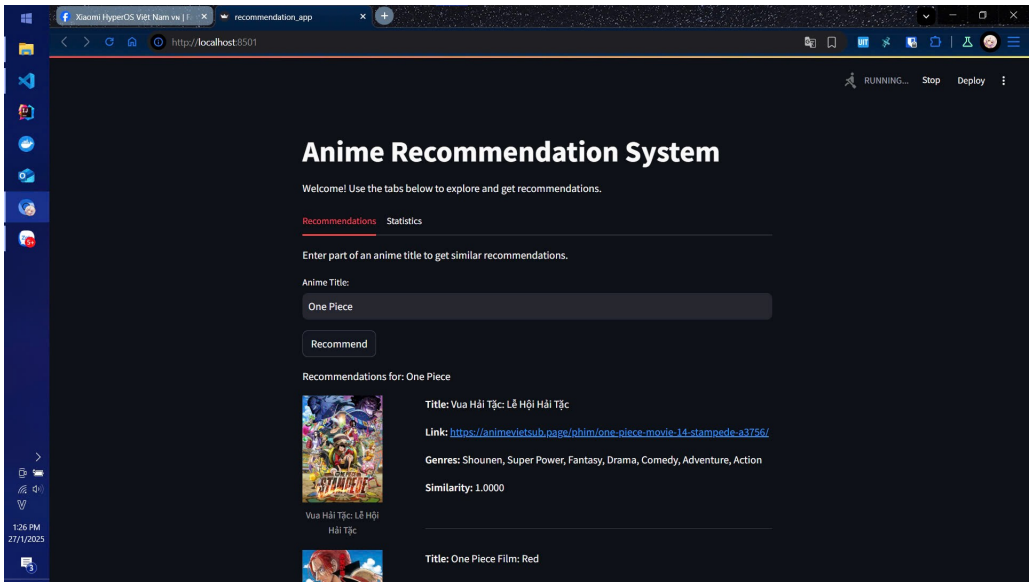
Below is the tab that records messages sent from Producer:



Hình 8: Messages Tab

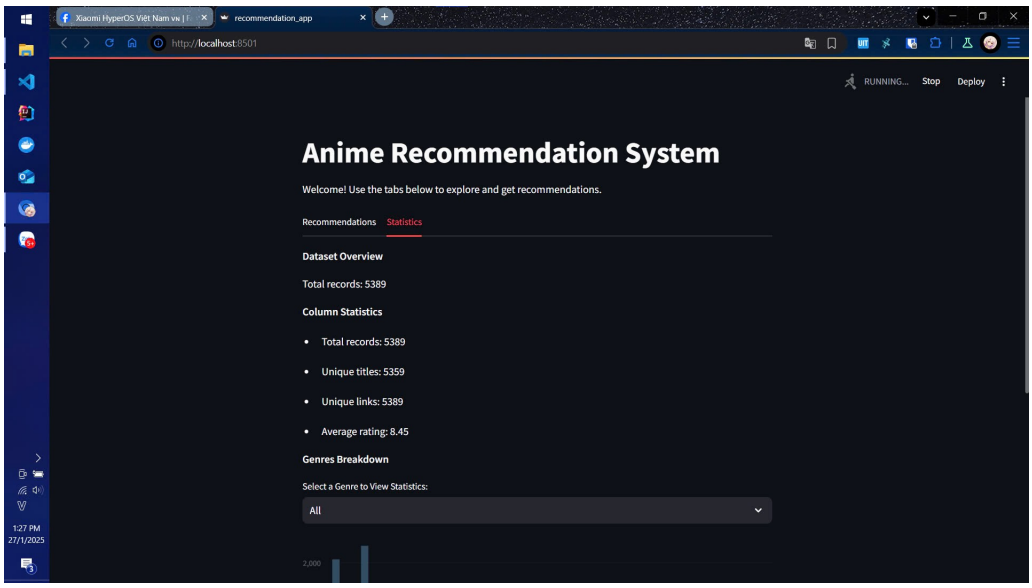
4.3 Simple web application using the Streamlit library

After starting the code, the result shows 2 tabs. One gives suggested results based on user's search, the other gives general statistics about the entire data. Below is the results when a user search for "One Piece":



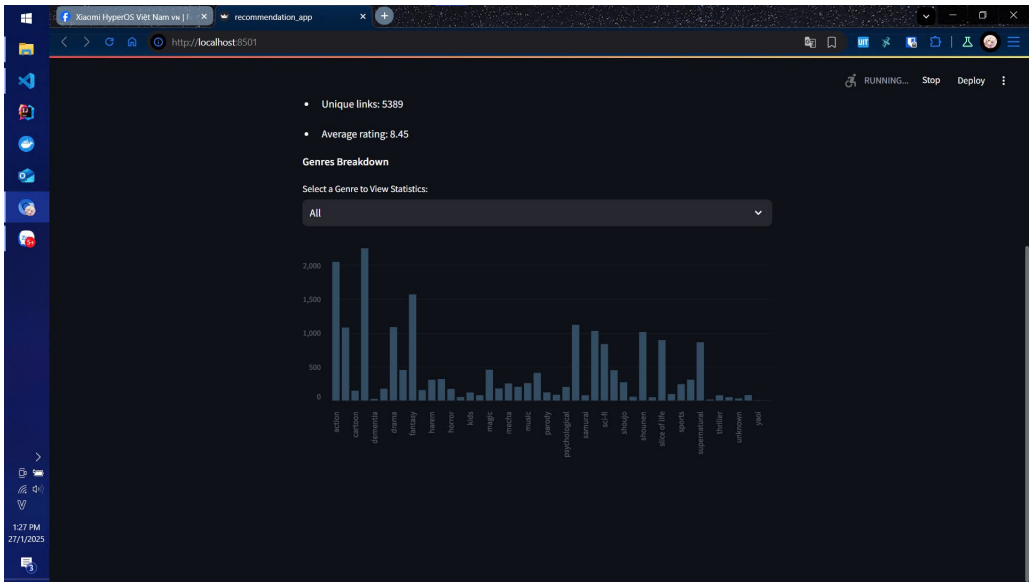
Hình 9: Deployment of the model

The tab below shows general statistical information about the data. There are a total of 5389 data fields and separate links but only 5359 titles are different.



Hình 10: Data statistics

The column chart below shows the amount of animes for each genres:



Hình 11: Chart showing the number of movies of each genre

5 Conclusion

In this project, the team developed and deployed a content-based anime recommender system utilizing PySpark and Kafka for efficient data processing and real-time recommendations. The system collects and processes anime data, including titles, genres, descriptions, and ratings, using Kafka for streamlining real-time data flow and PySpark for large-scale data processing. Through feature extraction and similarity measures, the content-based recommendation engine provides personalized anime suggestions tailored to user preferences.

The team found that using content-based filtering allowed for more accurate and relevant recommendations by focusing on the attributes of the anime rather than user interactions alone. This method ensures that users are presented with new anime options that closely match their interests, providing a valuable tool for anime enthusiasts looking to explore fresh content.

Looking ahead, the team plans to test the system's performance with different datasets, including user interaction data, to further enhance the recommendations through hybrid approaches. Additionally, efforts will be made to develop a user-friendly interface that integrates the recommendation system into an easily accessible web platform, providing users with seamless and interactive experiences for discovering anime content.

References

[1] K. A. Al-Rfou, B. S. Perozzi, & D. Skiena, *"DeepWalk: Online Learning of Social Representations"*. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.

[2] Y. Zhang, Y. Song, & M. Wang, *"Content-Based Collaborative Filtering for Recommender Systems"*. Journal of Computer Science and Technology, Volume 25, Issue 6, 2010.

[3] G. Adomavicius, & A. Tuzhilin, *"Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions"*. IEEE Transactions on Knowledge and Data Engineering, Volume 17, Issue 6, 2005.

[4] S. R. K. Manogaran, V. R. Shishira, & G. Shankar, *"Recommender Systems with Apache Spark and Hadoop"*. International Journal of Computer Applications, Volume 122, Issue 7, 2015.

[5] J. A. M. de Almeida, & G. C. Silva, *"Kafka for Real-Time Data Processing in Streaming Applications"*. International Journal of Computer Science and Information Technology, Volume 9, Issue 3, 2020.

[6] H. Zhang, Y. Li, & L. Chen, *"Content-Based Anime Recommendation with PySpark and Data Streams"*. Proceedings of the 2019 International Conference on Computer Science and Artificial Intelligence, 2019.

[7] P. B. S. A. Devi, K. M. Sharma, & S. K. P. P. Singh, *"Building a Content-Based Recommender System for Movies Using Spark and Machine Learning"*. Springer International Publishing, 2018.