



**MOHAMMED VI
POLYTECHNIC UNIVERSITY**

MASTER I QUANTITATIVE AND FINANCIAL MODELING
Option Data Engineering

SEMESTER 2 CAPSTONE PROJECT

Development of a Big Data Solution for News Recommendation

Student

Gbètoho Ezéchiél ADEDE

Supervisor

Pr Taoufik RACHAD

Requirement for validating the first year of the Master in
Quantitative and Financial Modeling

July 2024

Abstract

This project focuses on the development of a scalable news recommendation system aimed at delivering personalized news content to users based on their preferences. The system leverages real-time data processing and advanced machine learning techniques to categorize and recommend news articles efficiently. Key objectives include enhancing user engagement through intuitive interfaces, ensuring data persistence and retrieval for seamless user experiences, and automating workflows to streamline operations. The project emphasizes the importance of accurate news categorization and personalized recommendations to meet the diverse needs of users in consuming relevant and timely information.

Contents

Abstract	i
List of Tables	v
List of Figures	vii
1 General Introduction	1
2 Project Presentation	3
2.1 Introduction	3
2.2 Project Context	3
2.3 Problem	4
2.4 Proposed Solution Overview	5
2.5 Project Objectives	5
2.6 Conclusion	6
3 Methodology	9
3.1 Introduction	9
3.2 Data Source	9
3.2.1 Dataset Links	9
3.3 Preprocessing Each Dataset	10
3.3.1 Loading the Dataset	10
3.3.2 Dealing with Null Values	10
3.3.3 Feature Selection	10
3.4 Data Integration	10

3.4.1	Merging Datasets	10
3.4.2	Resolving Duplicates	11
3.4.3	Reducing the Categories	11
3.4.4	Removing Empty Descriptions	13
3.4.5	Feature Encoding	14
3.5	Data Augmentation	14
3.5.1	NLPAug Methods	14
3.6	Text Cleaning	15
3.6.1	Removing Special Characters	15
3.6.2	Text Transformation	15
3.6.3	Handling Empty Descriptions	15
3.7	Data Balancing	16
3.8	Saving the Preprocessed Data	16
3.9	Conclusion	16
4	News Categorization Model	17
4.1	Introduction	17
4.2	Approach and Methodology	17
4.2.1	Methodology Overview	17
4.2.2	Feature Engineering Methods	18
4.2.3	Machine Learning Algorithms	19
4.2.4	Evaluation Metrics and Settings	20
4.2.5	Machine/Spark Application Characteristics	20
4.3	Models Performance	20
4.4	Model Selection	21
4.5	Conclusion	22
5	Implementation of the Solution	23
5.1	Introduction	23
5.2	Data Pipeline Description	23

5.2.1	News Production	23
5.2.2	Metadata storage into Redis	24
5.2.3	Raw News Processing	24
5.2.4	Filtered News Preprocessing	25
5.2.5	Preprocessed News Processing	25
5.2.6	Filtered News Storage	25
5.2.7	Processed News Forwarding	26
5.2.8	Available News Recommendation	26
5.2.9	Interactions Publication	26
5.2.10	Interactions Storage	26
5.3	Automating the Data Pipeline with Airflow	27
5.3.1	Introduction to Airflow DAGs	27
5.3.2	DAGs Presentation	27
5.3.3	DAGs Schedule	30
5.4	System Architecture and Deployment Strategy	31
5.4.1	Kafka	32
5.4.2	Spark	34
5.4.3	Airflow	35
5.4.4	MongoDB	37
5.4.5	Redis	41
5.4.6	newsengine-client	42
5.5	Conclusion	45
6	General Conclusion and Future Work	47

List of Tables

3.1	Original and Grouped Categories	12
4.1	Summary of Models Performance	21
5.1	DAGs schedule	30
5.2	Kafka Topics Configuration	33

List of Figures

2.1	Simplified Architecture	5
5.1	News Production DAG	27
5.2	News ETL DAG	28
5.3	Interactions storage DAG	29
5.4	News Recommendation DAG	29
5.5	DAGs Schedule in Airflow	30
5.6	Automatic email sending with Airflow	31
5.7	Email example	31
5.8	Kafka Ecosystem	32
5.9	Kafka UI	33
5.10	Spark Cluster	34
5.11	Spark master	35
5.12	Spark connection in Airflow Admin	35
5.13	Celery Workers UI	36
5.14	News Recommendation Database	37
5.15	User Document	37
5.16	Filtered News Document	38
5.17	Interaction Document	39
5.18	Redis Databases	41
5.19	Registration Interface	43
5.20	OTP	43
5.21	User Login Interface	44

5.22 An example of News Article	44
5.23 User preferences	45

General Introduction

In today's digital age, the abundance of online news articles presents a challenge for users seeking personalized and relevant content tailored to their interests. Traditional methods of news consumption are giving way to digital platforms that offer curated news feeds based on user preferences. To meet this demand, the development of a robust news recommendation system becomes essential.

This project aims to design and implement a scalable big data-driven news recommendation system that is capable of efficiently handling large volumes of news data. By integrating real-time data processing capabilities, the system ensures prompt ingestion, categorization, and recommendation of news articles. The focus extends beyond accurate news categorization to include a user-friendly interface that enhances the overall browsing experience.

Key components of the system include advanced machine learning algorithms and big data technologies for news categorization, data preprocessing techniques for cleaning and augmenting news datasets, and the integration of automated workflows to manage the end-to-end data pipeline. The system architecture is designed for scalability and performance to accommodate increasing user demands.

Project Presentation

2.1 Introduction

The Project Presentation chapter provides an in-depth overview of the capstone project undertaken for the Master I in Data Engineering program at Mohammed VI Polytechnic University. This project aims to develop a robust and scalable big data solution for personalized news recommendation. The chapter outlines the context and motivation behind the project, the challenges posed by the current landscape of news consumption, and the innovative solution proposed to address these issues. Through a detailed exploration of the problem, an overview of the proposed solution, and the specific project objectives, this chapter sets the stage for understanding the comprehensive approach taken to design and implement the news recommendation system.

2.2 Project Context

This project is carried out as the capstone project for the second semester of the Master I program in Data Engineering at Mohammed VI Polytechnic University in Morocco. It serves as a culmination of all the knowledge and skills acquired throughout the academic year, encompassing various aspects of data engineering, including big data technologies, data processing, and analytics. The project spans two months, during which the primary objective is to apply theoretical concepts to a practical, real-world problem, thereby demonstrating proficiency in

designing and implementing a comprehensive data solution. In this context, the project aims to develop a big data solution for news recommendation, using technologies and methodologies learned during the course to address the challenges associated with large-scale data processing and personalized content delivery.

2.3 Problem

In the digital age, the vast amount of information available online has made it increasingly difficult for users to find relevant and personalized content. Traditional news platforms struggle to deliver personalized recommendations due to the sheer volume of articles published daily. This leads to an information overload, where users are either bombarded with irrelevant news or miss out on important content that matches their interests.

Moreover, the rapid pace at which news is generated requires a system that can process, filter, and analyze large volumes of data in real-time. Existing solutions often lack the scalability and efficiency needed to handle such tasks, resulting in delayed recommendations and suboptimal user experiences.

In addition, personalized news recommendation systems must address several technical challenges, such as:

- **Data Integration:** Aggregating news articles from various sources and ensuring the data is in a consistent and usable format.
- **Scalability:** Processing and analyzing large datasets efficiently to provide real-time recommendations.
- **User Personalization:** Accurately modeling user preferences and interactions to deliver relevant content.
- **Performance:** Ensuring that the recommendation system operates with low latency and high precision.

2.4 Proposed Solution Overview

The proposed solution employs a robust architecture leveraging modern big data technologies to handle real-time ingestion, processing, and recommendation of news articles. It integrates with external APIs to fetch diverse news sources, processes them through a streamlined pipeline, and utilizes advanced algorithms to deliver personalized news recommendations based on user preferences and interactions.

Figure 2.1 shows a simplified architecture of the proposed solution.

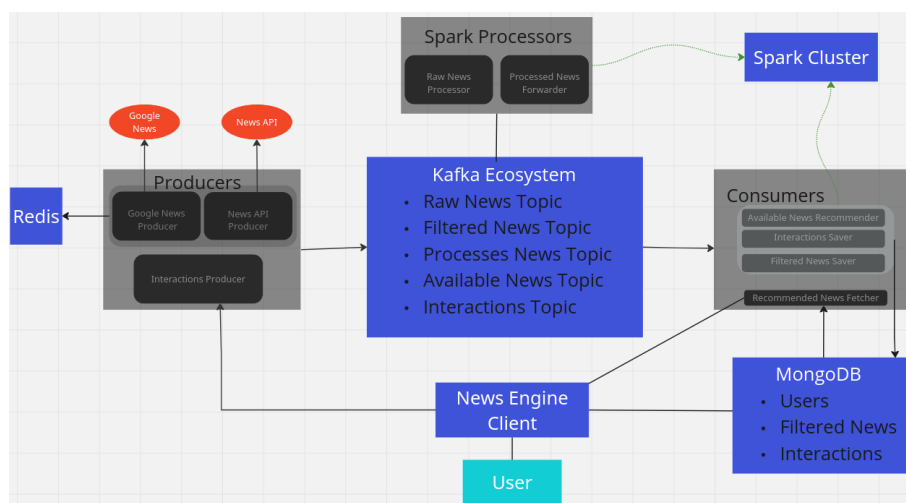


Figure 2.1: Simplified Architecture

2.5 Project Objectives

Develop a Scalable News Recommendation System

Create a system architecture that can efficiently handle the dynamic influx of news articles and user interactions, ensuring seamless scalability to accommodate future growth.

Implement Real-Time Data Processing

Design mechanisms for the ingestion, processing and analysis in real time of incoming news data, enabling instant updates and timely delivery of news recommendations to users.

Create a User-Friendly Interface

Develop an intuitive and engaging user interface that allows users to easily navigate through their personalized news feeds and interact with the recommended articles.

Ensure Data Persistence and Retrieval

Establish robust data storage strategies to persist news articles, user preferences, and interaction data, ensuring quick retrieval and efficient querying capabilities.

Automate Workflows

Set up automated processes to manage the entire data pipeline, from data collection and processing to the generation and delivery of news recommendations, minimizing the need for manual intervention.

Enhance Recommendation Accuracy

Develop and refine algorithms to improve the accuracy of news recommendations, ensuring that they are relevant and personalized to individual user preferences and behaviors, including categories, sentiments, and other user-specific criteria.

Deliver Personalized News Content

Ensure that users receive news articles adapted to their preferences, such as specific categories, sentiment analysis, and their history of seen, liked, and disliked news, enhancing the overall user experience and engagement.

2.6 Conclusion

This chapter has outlined the comprehensive approach taken to address the challenges of news recommendation in the digital age. By framing the context and objectives of the project, detailing the problem space, and presenting the proposed solutions, it provides a clear roadmap for the development and implementation of a scalable real-time news recommendation system. The

subsequent chapters will delve deeper into the technical implementation, showcasing how each component contributes to achieving the project's goals of delivering personalized and relevant news content to users.

Methodology

3.1 Introduction

This chapter details the preprocessing steps applied to the news data to prepare it for categorization in our news recommendation system. The process involves data loading, cleaning, transformation, integration, augmentation, and balancing to ensure the dataset is suitable for training machine learning models.

3.2 Data Source

The datasets are sourced from multiple CSV files and a JSON file containing news articles. These datasets include various fields such as title, description, category, and publication date. The data include four CSV files and one JSON file, collected from old news archives shared via Kaggle. The links to the dataset are shown below.

3.2.1 Dataset Links

1. [Topic-labeled news dataset](#)
2. [News categories dataset](#)
3. [Environment news dataset](#)

4. **Business and Financial news dataset**
5. **Indian financial news articles dataset**

3.3 Preprocessing Each Dataset

Each of the five collected datasets has been preprocessed. The following operations have been performed on them.

3.3.1 Loading the Dataset

The preprocessing begins by loading the news datasets into a Spark DataFrame. Spark is used to handle large datasets efficiently. The data is read from CSV or JSON files, ensuring the header information is included.

3.3.2 Dealing with Null Values

The dataset is checked for null values in the 'description' column, which is crucial for news categorization. Records with null descriptions are replaced by the title column if it exists and is not null. Otherwise, these records are removed to ensure data quality.

3.3.3 Feature Selection

Finally, only the relevant columns such as filtered description and news category have been selected.

3.4 Data Integration

3.4.1 Merging Datasets

The data integration begins with joining the five dataframes obtained in the previous step. The news datasets from different sources are merged into a single cohesive dataset. The merged dataset provides a comprehensive view of the news articles from various sources.

3.4.2 Resolving Duplicates

After merging, the dataset is checked for duplicate records. Duplicate entries are removed to ensure that each news article is unique and to avoid bias in the data analysis and modeling steps.

3.4.3 Reducing the Categories

The dataset was grouped by descending order of instances of each class, resulting in 48 classes initially. Some categories were further grouped to reduce the number of classes to 16.

The table [3.1](#) shows the original and grouped categories.

Original Categories	Grouped Categories
ARTS CULTURE & ARTS ARTS & CULTURE SPORTS COMEDY ENTERTAINMENT	ENTERTAINMENT-ARTS-CULTURE
MONEY BUSINESS BUSINESS & FINANCE	BUSINESS-ECONOMY-FINANCE
NATION U.S. NEWS WORLD NEWS THE WORLDPOST WORLDPOST WORLD	NATION & WORLD
SCIENCE TECH TECHNOLOGY	SCIENCE & TECHNOLOGY
COLLEGE EDUCATION	EDUCATION
HEALTHY LIVING HEALTH	HEALTH & WELLNESS
PARENTS PARENTING WEDDINGS DIVORCE	FAMILY & RELATIONSHIP
GREEN ENVIRONMENT	ENVIRONMENT
STYLE STYLE & BEAUTY HOME & LIVING	LIFESTYLE
TASTE FOOD & DRINK TRAVEL	TRAVEL & FOOD
BLACK VOICES QUEER VOICES WOMEN LATINO VOICES	SOCIAL ISSUES
WEIRD NEWS GOOD NEWS FIFTY	MISCELLANEOUS

Table 3.1: Original and Grouped Categories

After that, we removed classes that may lead to confusion, such as "MISCELLANEOUS", "SOCIAL ISSUES", "NATION & WORLD", "IMPACT", and finally, we got 12 categories of news:

1. BUSINESS-ECONOMY-FINANCE
2. ENTERTAINMENT-ARTS-CULTURE
3. ENVIRONMENT
4. HEALTH & WELLNESS
5. POLITICS
6. SCIENCE-TECHNOLOGY-MEDIA
7. FAMILY & RELATIONSHIP
8. TRAVEL & FOOD
9. LIFESTYLE
10. CRIME
11. RELIGION
12. EDUCATION

3.4.4 Removing Empty Descriptions

Any records with empty descriptions are removed from the dataset to ensure that all remaining records have meaningful content for analysis and modeling. This involves:

- Removing the rows with empty descriptions.
- Removing rows with probably non-meaningful descriptions, such as rows with less than 15 letters or 3 words.

3.4.5 Feature Encoding

Machine learning algorithms need numerical data, so we converted our category column to numerical values in the range [0, 11]. The mapping of numerical values to their categories was saved to a JSON file.

3.5 Data Augmentation

To enhance the dataset and have a fair number of instances of classes, data augmentation techniques are applied using NLPAug library. This involves generating synthetic data by slightly modifying the existing news descriptions while preserving their original meaning. Augmentation helps in increasing the diversity and size of the training dataset, making the models more robust.

3.5.1 NLPAug Methods

NLPAug supports the following methods of data augmentation:

- **Synonym Replacement:** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
- **Random Swap:** Randomly choose two adjacent words in the sentence and swap their positions.
- **Random Deletion:** Randomly remove each word in the sentence.
- **Antonym Replacement:** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its antonyms chosen at random.

The method we used is Synonym Replacement. As a word source to augment our data, we used WordNet.

3.6 Text Cleaning

3.6.1 Removing Special Characters

Special characters, punctuation, and extra spaces are removed from the descriptions to clean the text and maintain only alphanumeric characters. This step is essential for simplifying the text data and improving the performance of tokenization and other text processing techniques.

3.6.2 Text Transformation

Lowercasing

All text in the 'description' column is converted to lowercase. This step helps to reduce the complexity of text data by treating words with different cases as identical.

Tokenization

The cleaned lowercase text descriptions are split into individual words or tokens. Tokenization helps break down the text into manageable pieces for further processing.

Lemmatization

Lemmatization is applied to convert words to their base or root form. This step helps to normalize the text data by reducing different forms of a word to a single form, making it easier to analyze.

Stop Words Removal

Common stop words (e.g. 'and', 'the', 'is') that do not contribute significant meaning to the text are removed. This step helps to reduce noise in the data and focus on the more meaningful words.

3.6.3 Handling Empty Descriptions

After removing stop words, the dataset is checked for any empty or duplicate descriptions that may result from these transformations. The concerned rows are removed.

3.7 Data Balancing

The number of records in each news category is counted to identify, once again, any imbalances in the dataset. Although we have already applied data augmentation to balance our data, there may still be imbalances due to the previous removed rows. To address imbalances, the dataset is balanced by undersampling majority classes. This step ensures that the Machine Learning models trained on the data do not become biased towards any particular category.

This final version of our balanced news dataset contains:

- 65028 rows
- 12 categories of news
- 2 columns which are filtered description and numerical category

3.8 Saving the Preprocessed Data

The final balanced dataset is saved in both CSV and Parquet formats. These formats are chosen for their efficiency and compatibility with various data processing tools.

3.9 Conclusion

The preprocessing steps described above ensure that the news dataset is clean, consistent, and balanced, which makes it suitable for training robust news categorization models. These steps form a crucial part of the data preparation phase in the development of a news recommendation system.

News Categorization Model

4.1 Introduction

In this chapter, we detail the approach and methodology used to develop effective news categorization models. Our goal is to explore various feature engineering techniques and machine learning algorithms to classify news articles accurately. Specifically, we focus on feature extraction methods including HashingTF+IDF and Word2Vec embeddings, and evaluate their performance across a range of classifiers such as Naive Bayes, Linear Regression, Decision Trees, and Random Forests. Through rigorous evaluation and model selection based on performance metrics, we identify the most suitable approaches to accurately categorize news articles.

4.2 Approach and Methodology

4.2.1 Methodology Overview

Our methodology involves extracting meaningful features from the filtered descriptions obtained from the data preprocessing step, using HashingTF+IDF and Word2Vec embedding, and training multiple classifiers to categorize news articles. We emphasize model evaluation through rigorous train-test splitting, cross-validation, and parallel computing to ensure robust performance metrics.

4.2.2 Feature Engineering Methods

HashingTF+IDF (Term Frequency-Inverse Document Frequency)

Description:

- Converts text into numerical vectors and adjusts weights based on term frequency across documents.
- Efficient for handling large datasets and reducing feature space dimensionality.

Methodology:

1. **Explode the Filtered Descriptions:** Initially, the filtered descriptions were processed to extract individual words, which are essential units for subsequent feature extraction steps.
2. **Obtain Unique Words:** The next step involves obtaining unique words from the filtered descriptions, ensuring that each distinct word in the corpus contributes uniquely to the vocabulary.
3. **Calculate Vocabulary Size:** The vocabulary size, representing the total number of unique words in the filtered descriptions, was computed to understand the scope of the dataset's linguistic diversity.
4. **Determine Feature Space Dimensionality:** To determine the appropriate dimensionality of the feature space, the smallest integer n was calculated such that 2^n exceeds the vocabulary size. This step ensures that the feature space can adequately represent the vocabulary without excessive dimensionality.
5. **Define HashingTF and IDF Parameters:** With *numFeatures* determined, HashingTF was configured to transform the filtered descriptions into a numerical representation (*rawFeatures*). IDF was subsequently applied to adjust these features based on their importance across documents, yielding the final features used for model training and evaluation.

Each step in this process contributes to optimizing the representation of textual data for effective training of machine learning models. By carefully determining the *numFeatures* parameter and applying IDF, HashingTF+IDF provides a scalable and efficient method for feature engineering in text-based applications.

Word2Vec (Word Embeddings)

Description:

- Transforms words into dense vectors capturing semantic meanings based on co-occurrence in text.
- Useful for capturing context and semantic similarity in textual data.
- Best Vector Size: Optimal performance was consistently observed with a vector size of 300 dimensions across all machine learning algorithms tested (LR, DT, RF).

Methodology:

1. **Vector Size Experimentation:** Various vector sizes (100, 200, 300, 400) were evaluated to determine their impact on model performance.
2. **Performance Evaluation:** Models were trained and tested using each vector size across LR, DT, and RF algorithms.
3. **Observation:** Among the tested vector sizes, 300 dimensions consistently yielded the highest accuracy for all algorithms.

This approach ensures that Word2Vec embeddings effectively capture semantic nuances in the dataset, enhancing the performance of downstream classification tasks in news categorization.

4.2.3 Machine Learning Algorithms

- **Logistic Regression (LR):** Linear model for categorical outcome prediction.
- **Decision Trees (DT):** Nonlinear models using recursive splitting for interpretable classification.

- **Random Forests (RF):** Ensemble methods combining multiple trees to enhance accuracy and mitigate overfitting.

4.2.4 Evaluation Metrics and Settings

- **Train-Test Split:** 80% training and 20% testing to train models on majority data and assess real-world performance.
- **Cross-Validation (CV):** *numFold=3* for k-fold cross-validation to estimate model performance and reduce overfitting.
- **Evaluation Metric:** Accuracy was used as the primary metric to evaluate the performance of the model.
- **Parallelism:** *parallelism=3* in Spark environment for optimized computation during training and evaluation.

4.2.5 Machine/Spark Application Characteristics

- **Computing Environment:** Leveraged the SimLab supercomputer with 44 cores with 8 GB of RAM per core, i.e., 352 GB of RAM, on a GPU partition node, for efficient large-scale data processing.
- **Spark Application:** A local Spark application using all the 44 available cores and 320 GB of RAM.

4.3 Models Performance

Table 4.1 summarizes the performance metrics of various news categorization models using different feature engineering methods.

Feature Engineering Method	Model	Training Time	Train Accuracy (%)	Test Accuracy (%)
HashingTF+IDF	NB	16s	85	80
	LR	8m	96	80
	LR (tuned)	1h 27m	92	85
	DT	38m	18	18
	RF	4m	24	24
Word2Vec 300d	LR	19m	70	70
	LR tuned	21m	70	70
	DT	3m	37	37
	DT tunned	4h	81	57
	RF	3m	56	55
	RF tunned	6h	72	70

Table 4.1: Summary of Models Performance

4.4 Model Selection

Based on the performance metrics summarized in Table [Table 4.1](#), the model selection process considers both training and test accuracies across different feature engineering methods. For the **HashingTF+IDF** approach, the Logistic Regression (LR) model achieved the highest test accuracy of 85% after tuning, though it required substantial training time (1h 27m). In contrast, the Naive Bayes (NB) model trained relatively quickly (16s) but showed lower accuracy on both train and test sets. The decision tree (DT) and random forest (RF) models, despite their longer training times, struggled with accuracy, especially after tuning.

For the **Word2Vec 300d** embeddings, LR and RF models showed moderate performance, with LR achieving consistent results across both standard and tuned configurations. The decision tree models, however, exhibited varying degrees of accuracy, with significant improvements seen after tuning, albeit at the cost of longer training times. Overall, the LR model with Word2Vec embeddings stands out for its balanced performance in terms of accuracy and training efficiency.

Based on these findings, the LR model with HashingTF+IDF and Word2Vec embeddings emerges as the primary candidates for the news categorization task, balancing between computational efficiency and predictive performance.

4.5 Conclusion

In conclusion, our approach to developing news categorization models demonstrated the effectiveness of advanced feature engineering and machine learning techniques. By selecting the Logistic Regression model with Word2Vec embeddings, we achieved robust performance across different classifiers. Through careful selection of preprocessing methods and embeddings, coupled with thorough evaluation, we underscore the critical role of methodological rigor in achieving accurate and reliable text classification results. This lays a solid foundation for scalable and efficient news recommendation systems.

Implementation of the Solution

5.1 Introduction

The "Implementation of the Solution" chapter provides a comprehensive overview of the proposed news recommendation system. This system is designed to deliver personalized news articles to users based on their preferences and interactions. It encompasses various stages, including data collection, processing, and recommendation, utilizing a combination of advanced technologies such as Apache Kafka, Apache Spark, Apache Airflow, Redis and MongoDB. The solution is designed to be modular, scalable, and efficient, ensuring real-time processing and high-quality recommendations.

5.2 Data Pipeline Description

5.2.1 News Production

News collection: News articles are fetched daily from various sources such as NewsAPI and Google News, using predefined keywords.

News publication: These news articles are published to the RawNewsTopic Kafka topic into a standardized format including the following fields:

- The title of the news
- The publication date as a timestamp
- The description of the news
- The content of the news
- The source of the news
- The author of the news
- The URL of the news
- The URL of the image
- The identifier of the news

5.2.2 Metadata storage into Redis

Each news producer stores metadata into a Redis database. These metadata include:

- The datetime of the extraction as a timestamp
- The total number of news retrieved
- The maximum news id, which is stored or updated to be used later in combination with the producer identifier to generate a unique identifier for each news

5.2.3 Raw News Processing

Raw News extraction: The raw news published by the news producers are retrieved from the RawNewsTopic.

Raw News filtering: This step consists of filtering out:

- Articles with the same description, title, id, or URL
- Articles without title, description, content, or URL

Filtered news publication: Filtered news is then published to the FilteredNewsTopic.

5.2.4 Filtered News Preprocessing

Text Cleaning: Special characters, punctuations, and extra spaces are removed from the description column to keep only alphanumeric characters.

Text Transformation:

- Lowercasing: Convert the description column into lowercase.
- Tokenization: The cleaned text descriptions are split into individual words or tokens.
- Lemmatization: Convert words to their base or root form.
- Stop Words Removal: Remove common stop words (e.g., 'and', 'the', 'is') that do not contribute significant meaning to the text.

5.2.5 Preprocessed News Processing

Sentiment Analysis: Performed using an NLTK function, returning a score as a double value in $[-1,1]$. A column is added to specify whether the news is positive, negative, or neutral.

Categorization: An integer value in $[0,11]$ is returned as the prediction value, using the predictive model saved during the setup of the news categorization model. An additional column is added and represents the mapping of that prediction to the category name of the news.

Processed News Publication: Processed news articles are published to the ProcessedNewsTopic, with only the relevant columns.

5.2.6 Filtered News Storage

Filtered news messages are consumed from the FilteredNewsTopic and saved to a MongoDB collection `filtered_news`. This saved news can be used later to improve the performance of the news categorization model.

5.2.7 Processed News Forwarding

Processed news is forwarded from the `ProcessedNewsTopic` to the `AvailableNewsTopic` where it is ready to be recommended to users. Only the relevant columns are kept.

5.2.8 Available News Recommendation

Available news are recommended to each user based on their preferences for news categories and sentiments, their previous favorite or disliked news, and similarity with the news seen previously. The cosine similarity method is used to calculate the similarity between news articles.

Result: The ids of the recommended news for each user are stored in their profile in MongoDB. These ids will be retrieved, and the matching news will be consumed from the `AvailableNewsTopic` and displayed to each user once connected.

5.2.9 Interactions Publication

Likes, dislikes, and seen news articles are interactions of users with news articles. The interactions of the users with news articles are published in real time to a Kafka topic called `InteractionsTopic`.

- The user id
- The news id
- The datetime of the interaction
- The action: liked, disliked, or seen

5.2.10 Interactions Storage

Retrieving Kafka Messages:

- User Interactions: User interactions with news articles are retrieved from the `InteractionsTopic` Kafka topic.
- Available News: Available news articles, along with their processed features, are retrieved from the `ProcessedNewsTopic`.

Joining Data: The messages from the interactions topic and the processed news topic are joined on the news_id column.

Processing Data: The combined data is processed to remove duplicates, sort interactions by date, and select the most relevant interactions.

Storing Data in MongoDB: The processed interactions are converted into a suitable format and stored in the interactions collection in MongoDB. The user preferences are updated with the news action in the MongoDB users collection.

5.3 Automating the Data Pipeline with Airflow

5.3.1 Introduction to Airflow DAGs

DAG means Directed Acyclic Graph. A DAG is a Directed Acyclic Graph where tasks are nodes and the relations between these nodes are the dependencies between tasks. Airflow DAGs orchestrate and automate the various stages of the data pipeline. Each DAG is responsible for different tasks in the workflow, ensuring that data is processed, stored, and made available for recommendation.

5.3.2 DAGs Presentation

Four DAGs have been created.

DAG 1: News Production

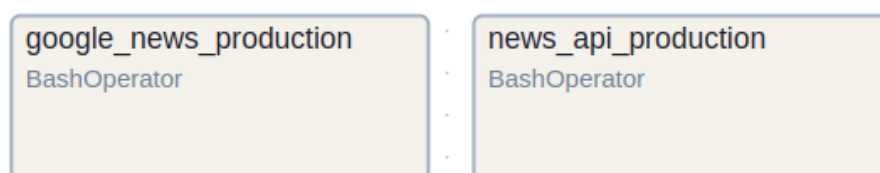


Figure 5.1: News Production DAG

Objective: To fetch news articles from multiple sources daily and publish them to the Kafka topic RawNewsTopic.

Tasks:

- News API Production Task: Collect news articles via Google NewsAPI, publish them on a standard format into RawNewsTopic, and store metadata into Redis.
- Google News Production Task: Collect news articles via GoogleNews API, publish them on a standard format into RawNewsTopic, and store metadata into Redis.

Dependencies: As shown in figure 5.1, there are no dependencies between the two tasks; they run in parallel.

DAG 2: News ETL (Extract, Transform, Load)

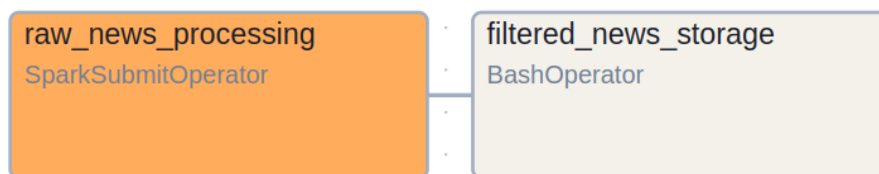


Figure 5.2: News ETL DAG

Objective: To process raw news articles.

Tasks:

- Raw News Processing Task: Retrieves raw news from the RawNewsTopic Kafka topic, filters out raw news, and publishes the filtered news into FilteredNewsTopic; preprocesses and processes the filtered news and publishes the processed news into ProcessedNewsTopic.
- Filtered News Storage Task: Consumes filtered news from the FilteredNewsTopic and saves them into a MongoDB collection (filtered_news) for further processing and storage.

Dependencies: As expected and as shown in figure 5.2, the filtered news storage task depends on the raw news processing task.

DAG 3: Interactions Storage

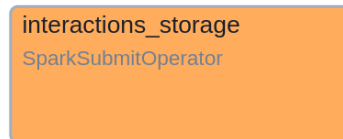


Figure 5.3: Interactions storage DAG

Objective: To store user interactions with news articles in MongoDB.

Task:

- Interactions Storage Task: Retrieve, process, and store user interactions, and update user preferences as described above.

DAG 4: News Recommendation

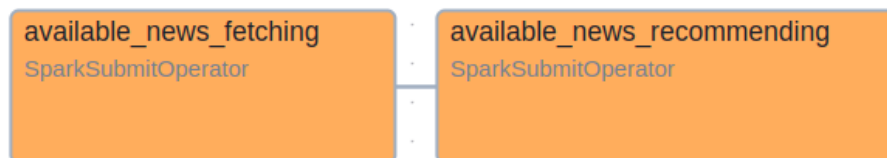


Figure 5.4: News Recommendation DAG

Objective: To generate and store news recommendations for users based on their preferences and interactions.

Tasks:

- Available News Fetching Task: Forwards processed news articles from the ProcessedNewsTopic to the AvailableNewsTopic, ensuring that only relevant news articles are made available for recommendation.

- Available News Recommendation Task: Generates personalized news recommendations for each user based on their interaction history and preferences, utilizing cosine similarity to calculate the similarity between news articles; stores the IDs of recommended news articles in each user's profile in MongoDB.

Dependencies: As expected and as shown in figure 5.4, the available news recommending task depends on the available news fetching task.

Let us mention that SparkSubmitOperator has been used to submit Spark jobs for tasks requiring Spark application.

5.3.3 DAGs Schedule

Table 5.1 and Figure 5.5 show the schedule of each of the four DAGs. Each DAG is executed every day at the time specified in the table.

DAG	Execution time
DAG 1: News Production	02:00 AM UTC
DAG 2: News ETL	03:00 AM UTC
DAG 3: Interactions Storage	04:00 AM UTC
DAG 4: News Recommendation	05:00 AM UTC

Table 5.1: DAGs schedule

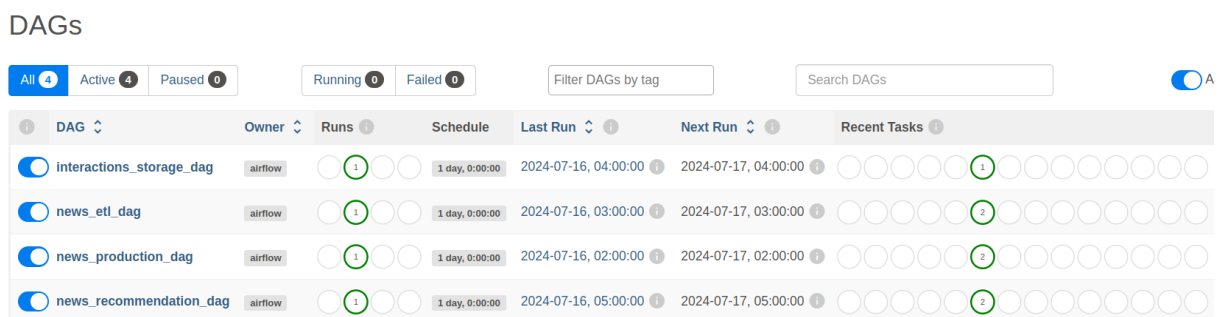


Figure 5.5: DAGs Schedule in Airflow

An email is automatically sent at the end of the execution of each task, regardless of whether it fails, retries, or succeeds, as can be seen in Figure 5.6 and Figure 5.8

bdnews.recommend	Airflow Task available_news_recommending - Success	Airflow Task Notification	Task ID: av...	6:00 AM
bdnews.recommend	Airflow Task available_news_fetching - Success	Airflow Task Notification	Task ID: available...	6:00 AM
bdnews.recommend 2	Airflow Task Interactions_storage - Success	Airflow Task Notification	Task ID: interactions_...	5:00 AM
bdnews.recommend	Airflow Task filtered_news_storage - Success	Airflow Task Notification	Task ID: filtered_ne...	4:02 AM
bdnews.recommend	Airflow Task raw_news_processing - Success	Airflow Task Notification	Task ID: raw_news_p...	4:02 AM
bdnews.recommend	Airflow Task google_news_production - Success	Airflow Task Notification	Task ID: google_...	3:00 AM
bdnews.recommend	Airflow Task news_api_production - Success	Airflow Task Notification	Task ID: news_api_pr...	3:00 AM

Figure 5.6: Automatic email sending with Airflow

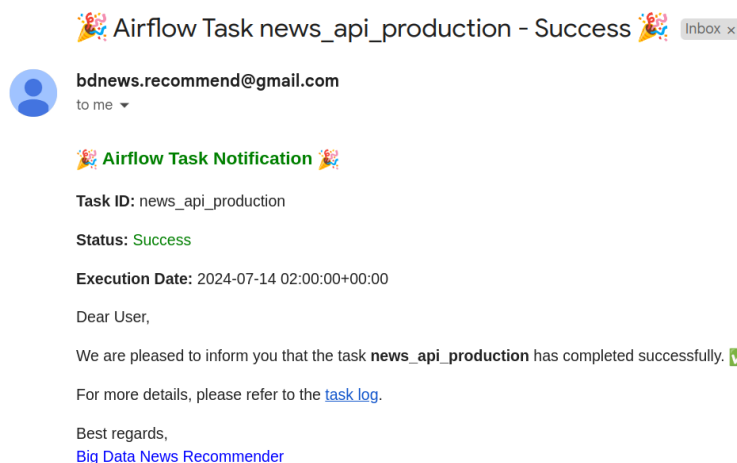


Figure 5.7: Email example

5.4 System Architecture and Deployment Strategy

The architecture of the news recommendation system is designed to be highly modular, scalable, and efficient, using various components to handle different tasks. This system is deployed using Docker and Docker Compose to ensure consistency across environments and simplify the deployment process.

Docker and Docker Compose

Docker is a platform that enables developers to package applications into containers—standardized executable components that combine application source code with the operating system libraries and dependencies required to run that code in any environment. Docker Compose is a tool that allows defining and running multi-container Docker applications. It uses a YAML file to configure the application's services, networks, and volumes.

The Docker Compose configuration for this system includes services for Kafka, Spark, Air-

flow, MongoDB, and Redis. Each service is defined in the ‘docker-compose.yml’ file, specifying the image to use, environment variables, volumes, and networks. The full configuration template can be found in the [Docker Compose template file on GitHub](#).

5.4.1 Kafka

Kafka is a distributed streaming platform that is used to build real-time data pipelines and streaming applications. Kafka brokers and topics handle the ingestion and processing of news articles and user interactions.

Kafka Components

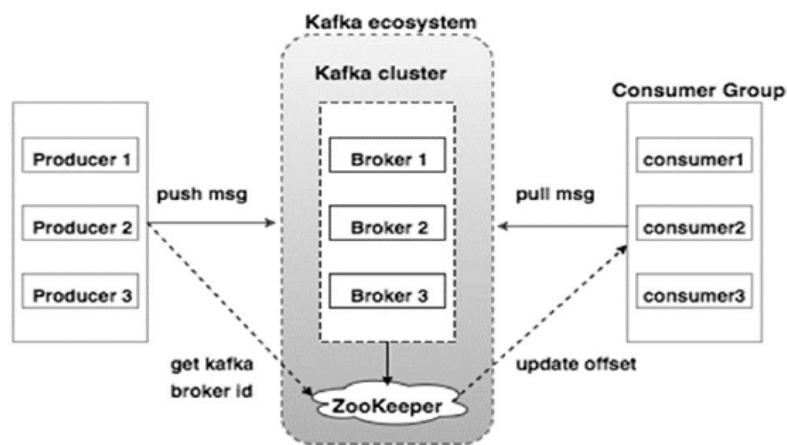


Figure 5.8: Kafka Ecosystem

- **zookeeper:** A service for managing Kafka brokers. Zookeeper has several functions:
 - Store metadata:
 - * about Kafka brokers, topics, and partitions.
 - * about which broker is the leader of which partitions and which brokers are followers of which partitions.
 - Coordinate partition leader election at start-up and reelection process in case of failure of one leader broker.
 - Coordinate communications between Kafka brokers.

- **kafka-broker1, kafka-broker2, kafka-broker3:**

Kafka brokers handle message streams. For each topic, one broker is a leader and the others are followers for each partition. The leader broker handles all read and write requests for the partition while the followers replicate the data. This setup ensures fault tolerance and data redundancy. If a leader broker fails, Zookeeper coordinates the election of a new leader from the available followers, ensuring the continuous availability and reliability of the message streams.

- **kafka-ui:** A web-based interface for managing Kafka, as shown in [Figure 5.9](#).

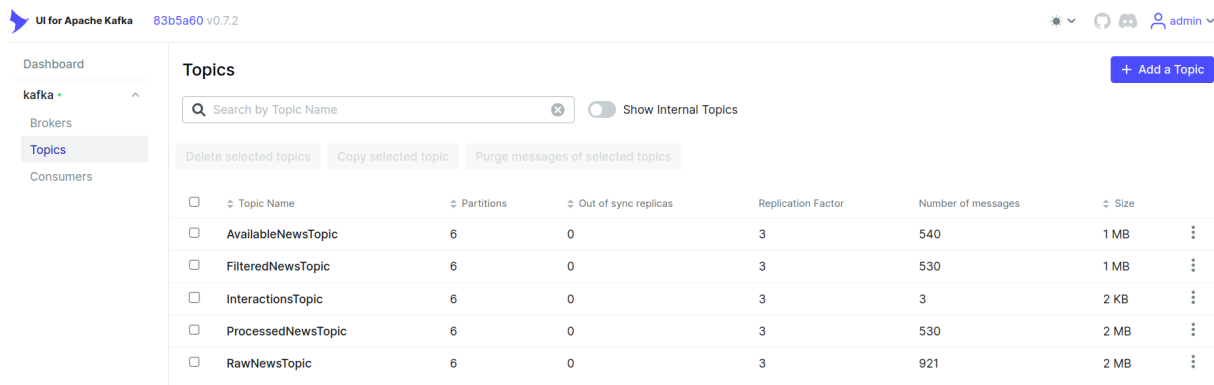


Figure 5.9: Kafka UI

Kafka Topics Configuration

The [Table 5.2](#) shows the configuration for each Kafka topic, including the number of partitions, the replication factor, and the retention duration.

Topic Name	Partitions	Replication Factor	Retention Duration
RawNewsTopic	6	3	2 hours
FilteredNewsTopic	6	3	2 hours
ProcessedNewsTopic	6	3	26 hours
AvailableNewsTopic	6	3	24 hours
InteractionsTopic	6	3	24 hours

Table 5.2: Kafka Topics Configuration

5.4.2 Spark

Spark is a unified analytics engine for large-scale data processing. Spark processes raw news articles, filters and transforms them, and generates recommendations.

Spark Components

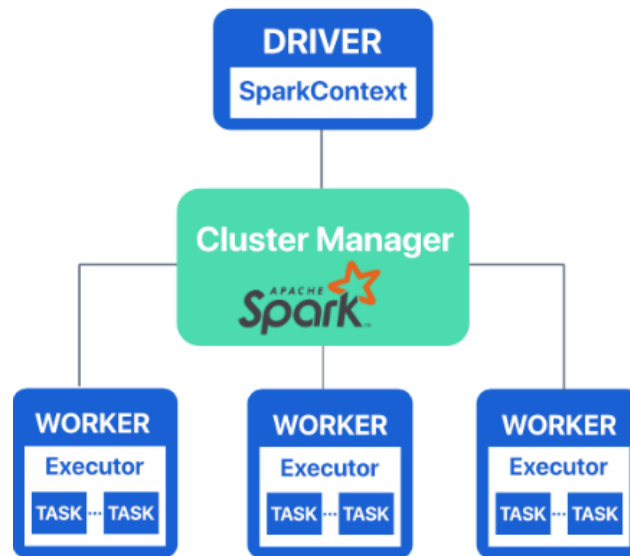


Figure 5.10: Spark Cluster

In this system, a Spark Standalone cluster is used to manage and execute jobs efficiently. A Spark Standalone cluster is a simple cluster manager included with Spark that makes it easy to set up and manage clusters. It consists of a master node and multiple worker nodes. The master node handles resource management and job scheduling, while the worker nodes execute the tasks assigned to them by the master.

- **spark-master:** The master node for Spark in a standalone cluster. It serves as the central coordinator responsible for resource management and job scheduling. The Spark master tracks the status of worker nodes and assigns tasks to them. The driver program, typically running on the master node, initiates the SparkContext. The SparkContext represents the entry point to Spark functionality, providing a connection to the cluster and managing the execution of tasks across the worker nodes.
- **spark-worker1, spark-worker2, spark-worker3:** Worker nodes for Spark, responsible

for executing tasks. These nodes communicate with the master node to receive tasks and report their status. In a Standalone cluster, each worker node can run multiple executors to parallelize task execution, providing high scalability and fault tolerance.

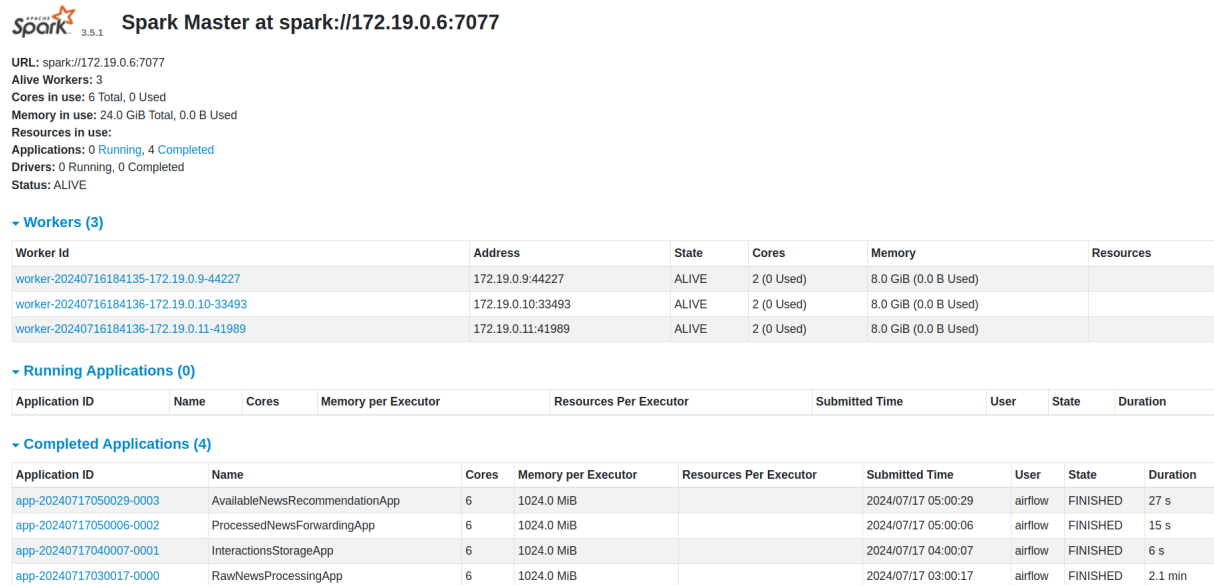


Figure 5.11: Spark master

To establish a connection with Spark to integrate large-scale data processing into the workflow orchestration, Airflow provides a seamless interface for managing Spark jobs. In the Airflow Admin interface (as shown in Figure 5.12), administrators can configure connections to Spark clusters. These connections define the necessary parameters such as the Spark master URL, authentication details, and other cluster-specific settings. Once established, Airflow leverages these connections to submit Spark jobs, monitor their execution, and manage dependencies between Spark tasks and other components in the data pipeline. This integration ensures efficient coordination and execution of Spark tasks within the broader workflow managed by Airflow.

Conn Id ↕	Conn Type ↕	Description ↕	Host ↕	Port ↕
spark-connection	spark	A connection to a Spark Standalone Cluster	spark://spark-master	7077

Figure 5.12: Spark connection in Airflow Admin

5.4.3 Airflow

Airflow is an open-source tool to programmatically author, schedule, and monitor workflows. Airflow manages the data pipelines, ensuring that tasks are executed in the correct order and at

the right time.

Airflow Components

Airflow consists of several key components that work together to facilitate workflow management:

- **airflow-init:** A one-time initialization service for setting up the Airflow database and creating the initial user.
- **airflow-webserver:** The Airflow web server for monitoring and managing workflows.
- **airflow-scheduler:** The Airflow scheduler for triggering tasks.
- **airflow-worker:** Celery workers for executing Airflow tasks.
- **airflow-flower:** A web-based tool for monitoring Celery workers.

The [Figure 5.13](#) shows a worker for an executed instance of the google news production task in the news production DAG.

Name	airflow.providers.celery.executors.celery_executor_utils.execute_command
UUID	ee6da301-12d9-430d-a8e3-5572424b1cef
State	SUCCESS
args	[['airflow', 'tasks', 'run', 'news_production_dag', 'google_news_production', 'scheduled__2024-07-14T02:00:00+00:00', '--local', '--subdir', 'DAGS_FOLDER/news_production_dag.py']]
kwargs	{}
Result	None
Received	2024-07-15 02:00:00.487093 UTC
Started	2024-07-15 02:00:00.489920 UTC
Succeeded	2024-07-15 02:00:31.100299 UTC
Retries	0
Worker	celery@709805d736d5
Timestamp	2024-07-15 02:00:31.100299 UTC
Runtime	30.611834363000526

Figure 5.13: Celery Workers UI

- **PostgreSQL:** The database backend used by Airflow to store metadata and manage task state.

5.4.4 MongoDB

```
news_recommendation_db> show collections
filtered_news
interactions
users
```

Figure 5.14: News Recommendation Database

MongoDB is a NoSQL database management system used to store user profiles, filtered news, and user interactions. A database named `news_recommendation_db` has been created with three collections for these purposes.

- `users`: Stores user profiles containing news preferences and recommended news.

```
news_recommendation_db> db.users.findOne()
{
  _id: ObjectId('669699daeb85f0757678b6b9'),
  firstname: 'Ezéchiel',
  lastname: 'ADEDE',
  email: 'ezechieladede@gmail.com',
  password: '399979ff6245cd927b9ee9e8eebcc79d24ef69a37314e53ccf49d82f5ef8fc16',
  creation_date: 1721145771,
  categories: [ 1, 4, 9 ],
  sentiments: [ 0, -1, 1 ],
  seen_news: [ { newsapi_251: -1 }, { newsapi_52: 1 }, { newsapi_711: 0 } ],
  ansa: true,
  recommended_news: [
    'newsapi_1279',
    'newsapi_251',
    'newsapi_820',
    'newsapi_825',
    'newsapi_379',
    'newsapi_1142',
    'newsapi_991',
    'newsapi_52',
    'newsapi_376',
    'newsapi_711'
  ]
}
```

Figure 5.15: User Document

It is important to note the following:

- `ansa` is a parameter used to indicate whether the user accepts news articles that lack a source or author.
 - `sentiments` is an array that holds a list of sentiment values: 0 for neutral, -1 for negative, and 1 for positive.
 - `seen_news` is an array of objects where each object contains `id` as the news ID and a value in {0, -1, 1} indicating the user's interaction with the news: 0 for seen, -1 for disliked, and 1 for liked.
- `filtered_news`: Stores news articles after filtering. This can be used later to improve the news categorization model.

```
news_recommendation_db> db.filtered_news.findOne()
{
  _id: 'newsapi_133',
  title: 'Vital Shiba Inu Metric Tumbles as SHIB Retraces - Bearish Phase Ahead?',
  description: 'Amid a broader market correction, Shiba Inu has witnessed a significant price decline, dropping 22% in the past week. SHIB traded at $0.00001596, with a 5.8% 24-hour decrease. Amid this...\n' +
    'The post Vital Shiba Inu Metric Tumbles as SHIB Retraces - Bearish Ph...',
  source_name: 'Techreport.com',
  url: 'https://techreport.com/crypto-news/vital-shiba-inu-metric-tumbles-as-shib-retraces-bearish-phase-ahead/',
  img_url: 'https://techreport.com/wp-content/uploads/2024/07/0_0-19.png',
  publication_date: 1720834855,
  lang: 'en',
  author: 'Rida Fatima'
}
```

Figure 5.16: Filtered News Document

- **interactions:** Stores user interactions with news articles, facilitating personalization and recommendation processes.

```
news_recommendation_db> db.interactions.findOne()
{
  _id: ObjectId('6695d8136a0f67dab60a7166'),
  news_id: 'newsapi_21',
  features: [
    131072,
    [
      9832, 11454, 32613,
      38751, 39471, 53472,
      59058, 64859, 67662,
      69512, 89833, 95391,
      112309, 124794, 128077
    ],
    [
      6.284122620767254, 8.628819199111426,
      6.766157094149232, 8.325136785313203,
      8.451888490952348, 7.213383703084036,
      7.283682910485043, 5.173894212415915,
      11.079467387080141, 6.206779827383694,
      6.0924252121676234, 19.885085734792963,
      6.710948270120506, 6.782992508613095,
      6.317668743258458
    ]
  ],
  date: 1721096104
}
```

Figure 5.17: Interaction Document

Features Field Explanation

The features field is essential for computing the similarity between different interactions and news articles, facilitating the recommendation process. Here's a more detailed breakdown:

- **Length of the Feature Vector (131072):** This number indicates that the original feature vector for the news articles has 131072 dimensions. It's a large, sparse vector typically used in high-dimensional spaces for machine learning tasks.
- **Indices of Significant Features:** This list contains the indices of the most significant features in the original feature vector. For example, the index 9832 indicates that the

feature at position 9832 in the original feature vector is significant for this interaction.

- **Values of Significant Features:** This list contains the actual values of the significant features. For example, the value 6.284122620767254 corresponds to the feature at index 9832. These values represent the importance or weight of the features in the interaction.

Use of the features Field in the Recommendation Process

The features field is crucial for computing the cosine similarity between different interactions and news articles. Here's how it's typically used:

- **Feature Extraction:** For each interaction, significant features are extracted and stored in the features field. This field comes from the feature engineering step during the processing of the description of the news article.
- **Similarity Computation:** When recommending news articles to a user, the system computes the cosine similarity between the feature vector of the news articles and the feature vectors of past interactions stored in the features field. Cosine similarity is a measure of similarity between two non-zero vectors, which calculates the cosine of the angle between them. It is defined as follows:

$$\text{cosine_similarity} = \frac{A \cdot B}{\|A\| \|B\|} \quad (5.1)$$

Where:

- * A and B are the feature vectors.
- * $A \cdot B$ is the dot product of the vectors.
- * $\|A\|$ and $\|B\|$ are the magnitudes (or Euclidean norms) of the vectors.

This calculation results in a similarity score between -1 and 1, where 1 indicates that the vectors are identical, 0 indicates orthogonality, and -1 indicates that the vectors are diametrically opposed.

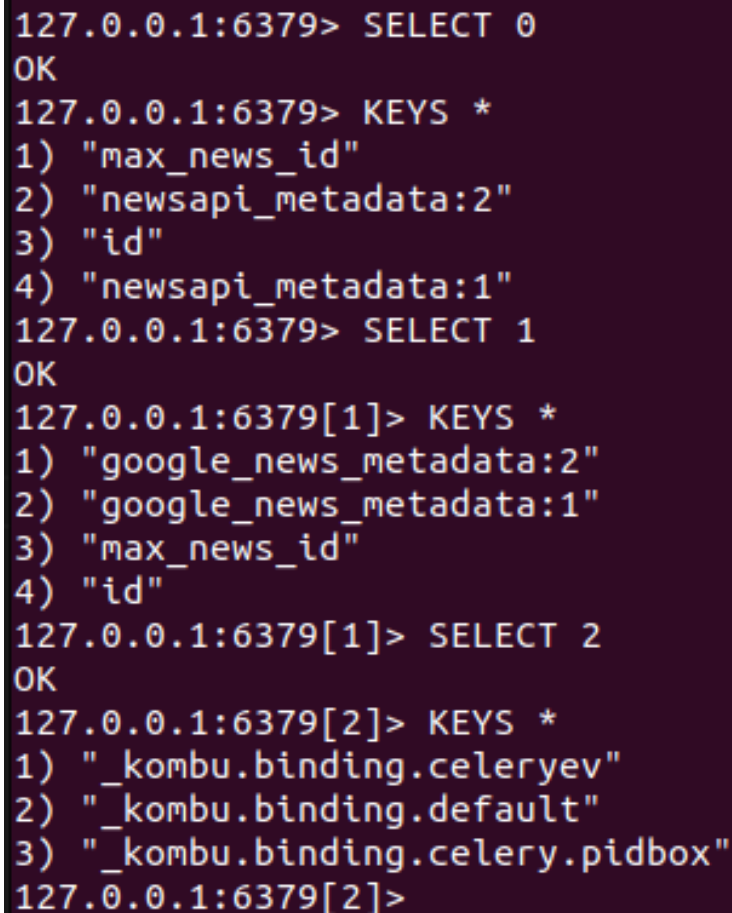
- **Personalized Recommendations:** Based on the computed similarity scores, the system identifies the most relevant news articles for the user and recommends them. Articles with higher similarity scores are more likely to be recommended because they closely match the user's past interactions.

5.4.5 Redis

Redis is an in-memory data structure store that is used to cache metadata.

As can be seen in [Figure 5.18](#), three databases have been used in Redis

- **db 0:** Stores metadata from the News API.
- **db 1:** Stores metadata from Google News.
- **db 2:** Used by Airflow Flower and Airflow workers for task management and metadata.



```
127.0.0.1:6379> SELECT 0
OK
127.0.0.1:6379> KEYS *
1) "max_news_id"
2) "newsapi_metadata:2"
3) "id"
4) "newsapi_metadata:1"
127.0.0.1:6379> SELECT 1
OK
127.0.0.1:6379[1]> KEYS *
1) "google_news_metadata:2"
2) "google_news_metadata:1"
3) "max_news_id"
4) "id"
127.0.0.1:6379[1]> SELECT 2
OK
127.0.0.1:6379[2]> KEYS *
1) "_kombu.binding.celeryev"
2) "_kombu.binding.default"
3) "_kombu.binding.celery.pidbox"
127.0.0.1:6379[2]>
```

Figure 5.18: Redis Databases

5.4.6 newengine-client

The ‘newengine-client’ component serves as the frontend interface for users to interact with the news recommendation system. It provides a user-friendly interface where users can:

- Login into their account
- View recommended news articles based on their preferences and interactions.
- Customize their news feed by adjusting preferences such as categories preferences and sentiment analysis settings.
- Interact with news articles by liking, disliking, or marking them as seen to improve future recommendations.

The client interface is designed to be responsive and intuitive, ensuring a seamless user experience across different devices and platforms. It leverages modern web technologies to deliver dynamic content and personalized recommendations effectively.

Implementation with Streamlit

The ‘newengine-client’ interface is implemented using Streamlit, a modern framework for building web applications with Python. Streamlit allows for rapid prototyping and development of interactive web applications directly from Python scripts. It integrates seamlessly with data processing and recommendation algorithms implemented in the backend, ensuring real-time updates and personalized content delivery to users.

Integration with backend services such as Kafka, Spark, Airflow, MongoDB, and Redis ensures that the client receives up-to-date and relevant news recommendations in real-time. This component plays a crucial role in bridging the gap between users and the underlying data processing and recommendation engine.

Illustrations

The following figures illustrate some interfaces.


● Register


Register

Firstname
Ezéchiél

Lastname
ADEDE Press Enter to submit form

Email
ezechieladede@gmail.com

Password
..... 

Confirm password
..... Press Enter to submit form 

Register

Figure 5.19: Registration Interface

Register

Thank you, Ezéchiél! A 6-digit confirmation code has been sent to your email address.

Enter the 6-digit confirmation code
948287 Press Enter to submit form

Verify Code

Figure 5.20: OTP

Welcome to Big Data News Recommendation App

Get news articles based on your preferences

Welcome to the Big Data News Recommendation App! Here, you'll find the most relevant and up-to-date news articles tailored to your interests using cutting-edge big data technology. Register or log in to start receiving personalized news recommendations right away!

Choose an option:

- ☒ Login
☐ Register

Login

Email

Password

Login

Figure 5.21: User Login Interface

ENTERTAINMENT-ARTS-CULTURE



Today's WNBA Games: Saturday - TV Channel, Game Times and Stats

USA Today | Data Skrive 2d



0.886

Figure 5.22: An example of News Article

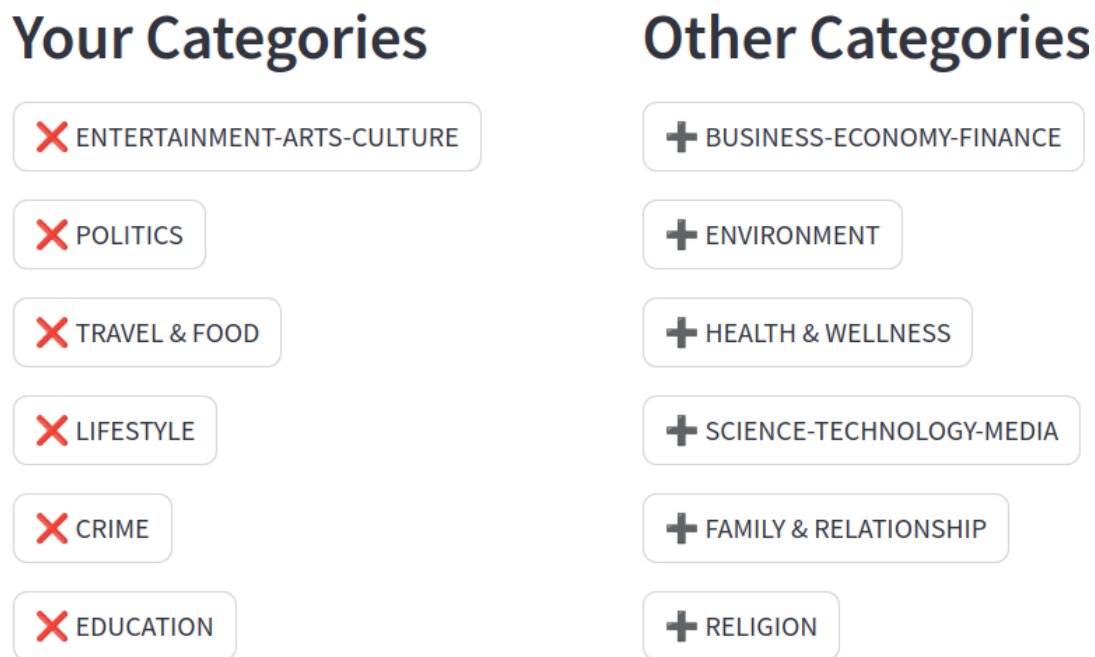


Figure 5.23: User preferences

5.5 Conclusion

In this chapter, we have discussed the detailed implementation of our news recommendation system. By integrating Apache Kafka for real-time data streaming, Apache Spark for data processing, and Apache Airflow for workflow orchestration, we have created a robust framework capable of handling large-scale data processing and personalized content recommendations.

The use of Docker and Docker Compose has facilitated seamless deployment and management of our system across different environments. This approach ensures consistency and scalability, allowing us to efficiently manage resources and adapt to varying workloads.

Overall, the implementation demonstrates our commitment to leveraging modern technologies to deliver a reliable and scalable news recommendation system. With a strong foundation in place, we look forward to exploring new opportunities for innovation and improving our system's capabilities in the future.

General Conclusion and Future Work

The news recommendation system presented in this report demonstrates a robust, scalable, and efficient approach to delivering personalized news content to users. Leveraging advanced technologies such as Apache Kafka, Apache Spark, Redis, MongoDB, and Airflow, the system effectively manages the entire data pipeline from collection to recommendation. Its modular design ensures flexibility and ease of maintenance, while Docker and Docker Compose enable consistent deployment across diverse environments. This comprehensive solution not only enhances user experience by delivering timely and relevant news but also lays a solid foundation for future improvements and scalability.

Throughout this project, we have showcased the potential of integrating various big data technologies to address complex real-time data processing challenges. Looking ahead, several areas offer opportunities for enhancement and expansion:

Future Work

- **Integration of Additional News Producer APIs:** Expand the system's capability to ingest news from multiple sources by integrating additional News Producer APIs. This diversification can enrich the content available for recommendation and improve coverage across different domains.
- **Enhancement of News Categorization Model:** Improve the accuracy and granularity of

news categorization by adopting advanced machine learning techniques. Techniques like stacking and deep learning models could be explored to better classify news articles based on topics, sentiment, and relevance to user preferences.

- **Enhancement of User Interface (UI) and User Experience (UX):** Continuously refine the newsengine-client interface using Streamlit to improve usability, accessibility, and interactivity. Incorporate user feedback mechanisms to gather insights for further UI/UX enhancements.
- **Deployment and Scaling Strategies:** Explore advanced deployment strategies such as Kubernetes for orchestration and scaling of containerized components. Implement auto-scaling mechanisms to handle fluctuations in user traffic and data volumes effectively.
- **Multilingual Support:** Currently, English is the only one supported language. Future work will focus on adding support for multiple languages to cater to a diverse user base and provide personalized news recommendations in their preferred language.
- **Optimized Resource Configurations:** Investigate and apply the optimal resource configurations for key components such as Kafka and Spark. This includes fine-tuning memory, CPU, and other resource allocations to enhance performance, reliability, and efficiency of the system.

By pursuing these avenues for future work, the news recommendation system can evolve into a more intelligent, adaptive, and user-centric platform. These enhancements not only align with current industry trends but also position the system to meet the evolving demands of personalized content delivery in the digital age.

Additional Resources

For more information on the implementation and deployment of the news recommendation system, please refer to the project's GitHub repository: <https://github.com/Starias22/Big-Data-News-Recommender>.