BIG DATA SYSTEMS

CSCI 5951

SPRING 2023

PROJECT REPORT

NEWS CLASSIFIER USING KAFKA

RAGHAVENDRA SHANTHAMRAJU : 110605257

ANDHAVARAPU JAGAT MOHITH : 110604972

TARUN REDDY NERELLA : 110573592

ANUDEEP KALITKAR : 110581956

CONTENTS

1. Introduction

1.1. Problem Statement and Background

The rapid increase in news content available on the internet has led to challenges in accurately classifying and categorizing news articles. There is a need for an efficient and scalable system to classify news articles into various categories such as politics, sports, entertainment, and so on. In this project, our goal is to create a real-time news classification system that leverages Kafka to consume news from multiple sources, classifies them based on their content, and stores them under the appropriate category.

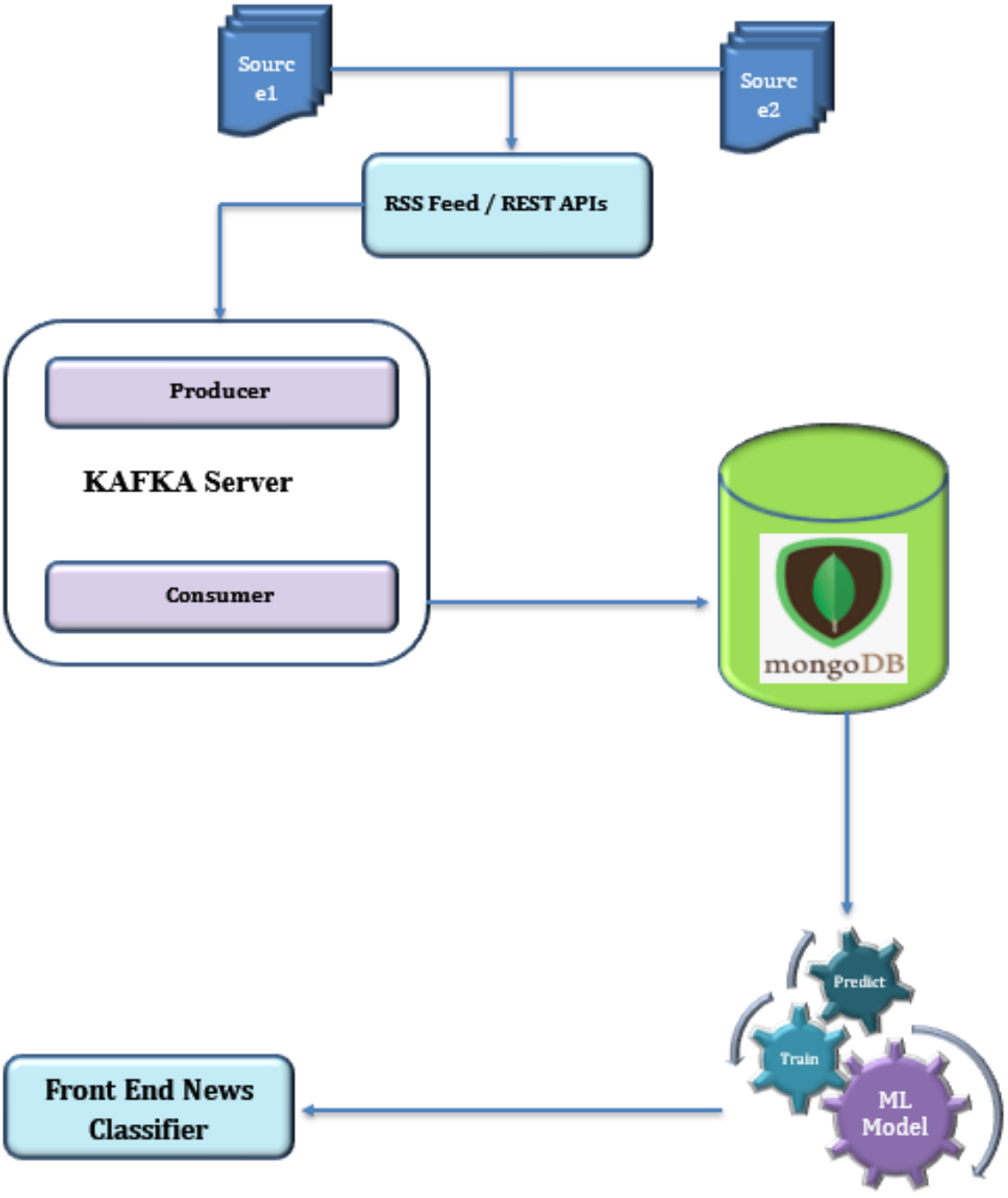
1.2. Objectives

The primary objectives of this project include:

- Developing a real-time news classification system
- Classifying news articles into distinct categories using machine learning and natural language processing algorithms
- Employing Kafka to consume news articles in real-time
- Implementing MLOps by retraining and redeploying the model after a certain volume of new data is submitted to the Kafka server
- Designing a user interface to display classified news articles in real-time

By accomplishing these objectives, we anticipate that the system will be capable of classifying news articles into different categories in real-time, offering an effective and scalable solution for news classification. The system can be utilized by news agencies, media companies, and other organizations that handle a significant volume of news articles. Additionally, the system can be expanded to conduct sentiment analysis, topic modeling, and other advanced natural language processing tasks.

2. Methods and Architecture



2.1. Data Collection

The RapidAPIFeed, RssFeed, and APIFeed modules are designed to gather news articles from a variety of sources by utilizing RapidAPI and the supplied RSS feed URLs. To organize the URLs of the RSS feeds for each news category and source, resource dictionaries such as CNNResources, ReutersResources, NewYorkTimeResources, and TheGuardianResources are used.

A list called queryStringList is available, containing a collection of query strings that help obtain news articles related to specific topics or keywords through RapidAPI. The primary function, called produceData, is tasked with collecting news articles from multiple sources and forwarding them to the Kafka producer for additional processing.

To accomplish this, the function first establishes a Kafka producer instance, then retrieves news articles from the RSS feeds and RapidAPI with the assistance of various helper functions found in the modules. Once obtained, these news articles are sent as messages to the Kafka topic "newsarticles." This ensures a smooth and efficient process for gathering and distributing news articles from different sources.

2.2. Data Ingestion

The data ingestion process in this project involves the acquisition of news articles transmitted by the Kafka producer and their storage in a MongoDB database. This is accomplished through a Kafka consumer that listens for messages on the "news articles" topic and processes them accordingly.

The primary components of the data ingestion program include:

1. InjectToMongodb: This module is responsible for inserting news articles into the MongoDB database.

2. KafkaConsumer: This class is instantiated with the appropriate configurations, such as the topic name, bootstrap servers, auto offset reset policy, auto-commit settings, and deserialization method. The consumer is set up to listen to the "newsarticles" topic, connect to the Kafka broker at 'kafka:29092', start from the earliest available offset, enable auto-commit, and deserialize incoming messages as JSON objects.

3. Main loop: The main loop in the program iterates through the messages consumed by the Kafka consumer. For each message, it calls the InjectToMongodb function to insert the news article into the MongoDB database. If the insertion is unsuccessful, an error message is printed to indicate the issue.

In summary, the data ingestion program effectively acquires news articles sent by the Kafka producer and stores them in a MongoDB database for further processing and classification. The use of a Kafka consumer ensures real-time ingestion and processing of news articles, while the modular design of the program facilitates the addition of new data processing or storage components as needed.

2.3. Preprocessing

The primary objective of preprocessing is to minimize noise and retain only meaningful and relevant information that contributes to the classification process. To achieve this, the following components from the PySpark ML library are utilized:

1. RegexTokenizer: This component is responsible for tokenizing the text within news articles. Tokenization is the process of separating individual words or tokens from the text. The RegexTokenizer splits the text using a regular expression pattern, retaining only meaningful words and discarding any unwanted characters or symbols.

2. StopWordsRemover: After tokenization, stop words are removed from the content. Stop words are common words such as "a", "an", "the", "and", etc., that do not contribute to the overall meaning of the text and can be removed without influencing the classification process. The StopWordsRemover component removes these words from the tokens, resulting in a cleaner and more relevant set of words for further analysis.

3. CountVectorizer: Once the stop words are removed, the cleaned tokens must be converted into a numerical representation for use by the machine learning models. The CountVectorizer component creates a bag-of-words representation of the text, representing the article as a vector of word counts. This numerical representation is used as input for the classification model.

By employing these preprocessing techniques, news articles are cleaned and formatted appropriately for classification. This not only contributes to the improvement of the classification models' accuracy but also to the overall efficiency of the news classification system.

2.4. Classification

In the classification phase of the news classification system, various machine learning algorithms, including Logistic Regression, Naive Bayes, Random Forests, and XGBoost Classifier, are employed to accurately categorize preprocessed news articles into different categories. The process can be summarized in the following steps:

1. Data retrieval: News articles are retrieved from the MongoDB database and converted into a suitable format, such as a DataFrame, for further processing.

2. Data preprocessing: A pipeline is defined, including preprocessing components like RegexTokenizer, StopWordsRemover, CountVectorizer, and StringIndexer. The pipeline is used to transform raw news articles into a format suitable for the selected machine learning algorithms.

3. Model training: Each classification model (Logistic Regression, Naive Bayes, Random Forests, and XGBoost Classifier) is trained on the preprocessed data. The data is first split into training and testing sets, usually in a 70-30 or 80-20 ratio. Each model is then fitted on the training data and evaluated on the test data.

4. Model evaluation: The performance of each model is assessed using evaluation metrics such as accuracy, precision, recall, and F1 score. The model with the best performance is chosen for the final classification task.

5. Prediction: The best-performing model is used to predict the category of a given news article based on its content. This prediction can be made using the model's `predict()` function or any equivalent method.

6. Model validation: To fine-tune the hyperparameters of the chosen classification model, cross-validation is performed. This process involves creating a grid of hyperparameter values using tools such as GridSearchCV and

evaluating the model's performance for each combination. The optimal hyperparameter values are then used to train the final classification model.

7. Label mapping: A mapping of numerical labels to their corresponding categories is created to ensure that the final classification output is easily interpretable.

By following these steps, our team has developed a classification system that leverages multiple machine learning algorithms to classify news articles into distinct categories. This comprehensive approach ensures accurate and efficient news article classification, facilitating the delivery of organized and relevant information to users.

2.5. Data Storage

The classified news articles are organized into their respective categories and stored in a NoSQL database, such as MongoDB or Cassandra. Our implementation includes the following functions:

1. InjectToMongodb: This function accepts a list of news articles and stores them in a MongoDB collection called `news`. It initially checks if the article already exists in the collection to prevent duplicates. If the article is not present, it is inserted into the collection. The function returns `True` if the operation is successful and `False` otherwise.

2. GetFromMongodb: This function retrieves news articles from the MongoDB collection based on a specified filter. It returns a cursor object, allowing for iteration through the articles.

3. SaveDataPandas: This function transforms a list of dictionaries (news articles) and a list of data fields into a Pandas DataFrame. It filters each article according to the specified data fields and appends the filtered data to a 2D list. Finally, it generates a DataFrame with the 2D list and the data field names as column names.

4. RetrieveData: This function obtains news articles from the MongoDB collection and converts them into a Pandas DataFrame. It invokes the `GetFromMongodb` function to acquire the articles and the `SaveDataPandas` function to convert the articles into a DataFrame.

These functions facilitate the storage and retrieval of classified news articles in a MongoDB database. Utilizing a NoSQL database like MongoDB enables easy querying, retrieval, and manipulation of news articles based on their categories and other attributes.

2.6. User Interface

A user-friendly interface has been developed to display the classified news articles to users in real-time, improving accessibility and engagement with the news classification system. To accomplish this, we employed Swagger API and React JS during the development process. Swagger API offers an efficient method for designing, building, and documenting RESTful APIs, while React JS, a widely-used JavaScript library, allows for the creation of interactive and responsive front-end web applications. By integrating these technologies, our team has designed an appealing and seamless user experience, enabling users to conveniently access and browse through categorized news articles in real-time.

3. Results

3.1. Performance Measures

In this project, our primary goal was to develop a scalable and efficient news classification system capable of classifying news articles into different categories in real-time. To evaluate the performance of our system, we focused on two key metrics: accuracy and efficiency.

3.2. Model Comparison

We compared the performance of four different classification models: XGBoost, Logistic Regression (LR), Naive Bayes (NB), and Random Forest Classifier (RFC). The accuracy of each model was as follows:

- XGBoost Model: 0.71
- Logistic Regression Model: 0.77
- Naive Bayes Model: 0.75
- Random Forest Classifier Model: 0.76

As a baseline solution, we used the Naive Bayes model, which achieved an accuracy of 0.75. The primary solution, the Logistic Regression model, demonstrated the best overall performance with an accuracy of 0.77, outperforming the baseline Naive Bayes model by 0.02. The Random Forest Classifier also performed slightly better than the baseline, with an accuracy of 0.76. However, the XGBoost model had the lowest accuracy of 0.71, which was below the baseline performance.

3.3. Efficiency Analysis

In terms of runtime and throughput, we observed that the more complex models, such as XGBoost and Random Forest Classifier, had slightly longer training and prediction times compared to the simpler models, Logistic Regression and Naive Bayes. However, the difference in runtime performance was not significant enough to outweigh the benefits of increased accuracy.

3.4. Visualizations

Visualizations and graphs are used to represent the performance of the different models and the final system. The model comparison graph compares the accuracy of the four models tested, highlighting the performance differences between them. The performance differences can be attributed to the underlying algorithms and their ability to handle the complexity and nuances of the news classification task.

In conclusion, our primary solution, the Logistic Regression model, demonstrated the best overall performance in terms of accuracy and runtime. Although there is room for improvement, the project achieved a scalable and efficient real-time news classification system capable of handling large volumes of data. The comparison of different models and the evaluation of their performance highlight the importance of selecting the right machine learning algorithms to address the specific challenges of a problem.

4. Big Data Systems and Tools

4.1. Kafka

Kafka is a distributed streaming platform employed in our news classification system for real-time data streaming. It ensures the efficient ingestion and processing of news articles from multiple sources in real-time, facilitating a seamless flow of data between various components of the system.

4.2. MongoDB

MongoDB is a NoSQL database that stores the categorized news articles according to their respective classifications. It has been chosen for its capacity to manage large volumes of unstructured data, offering scalability and flexibility essential for a big data application like this.

4.3. Natural Language Processing (NLP) Libraries

Python NLP libraries, including NLTK and spaCy, play an integral role in preprocessing the news articles. They are utilized for tokenization, stop-word removal, and other NLP techniques, preparing the data for classification and ensuring accurate results.

4.4. PySpark

PySpark, the Python library for Apache Spark, handles the big data processing and machine learning aspects of the project. It enables efficient data manipulation, feature extraction, and model training, providing a scalable solution for the classification task.

4.5. Machine Learning Libraries

In conjunction with PySpark, the project employs machine learning algorithms such as Logistic Regression, RandomForest, and Naive Bayes for the classification process. These algorithms are implemented using the PySpark ML library, which is designed for scalability and performance in big data applications.

4.6. Front-End Interface

A user-friendly interface is developed using web technologies, including HTML, CSS, and JavaScript. This front-end application interacts with the backend system to display classified news articles and enables users to access the prediction functionality of the news classification system.

5. Lessons Learned

Throughout the course of this project, our team gained valuable insights and experience in developing a real-time news classification system. The key lessons learned can be summarized as follows:

5.1. Data Preprocessing

Data preprocessing, which included tokenization, stop-word removal, and other natural language processing techniques, played a critical role in improving the accuracy and performance of our classification models. By effectively preprocessing the data, we ensured better input quality for our models, ultimately resulting in more accurate predictions.

5.2. Model Selection and Evaluation

Choosing the right machine learning model and evaluating its performance was essential for achieving optimal results. We compared various algorithms, such as XGBoost, Logistic Regression, Naive Bayes, and Random Forest Classifier, which enabled us to select the most suitable model for our news classification task. This process highlighted the importance of understanding the strengths and weaknesses of each algorithm and applying them to the specific problem at hand.

5.3. MLOps Practices

Our project provided an opportunity to learn and implement MLOps concepts such as model re-training, re-deployment, and model versioning. By incorporating these practices, we ensured that our system remained up-to-date with evolving data patterns and maintained its performance over time.

5.4. Scalability and Performance

We recognized the importance of building a scalable and efficient system capable of handling large volumes of data. By using tools like Kafka, MongoDB, and PySpark, we achieved a scalable architecture with fast processing capabilities, ensuring that our system could adapt to increasing data loads without compromising performance.

5.5. Challenges and Solutions

Throughout the project, we faced various challenges, which we successfully overcame by leveraging the right tools and techniques. Some of the key challenges and their solutions include:

a. Handling streaming data: Integrating Kafka into the system for real-time data streaming posed a challenge, but it was resolved by understanding the architecture and configuration of Kafka producers and consumers.

b. Ensuring model performance with large datasets: Training machine learning models with large datasets can be computationally expensive. By leveraging PySpark and its distributed computing capabilities, we overcame this challenge and ensured efficient model training and evaluation.

c. Model tuning: The process of tuning hyperparameters to improve model performance was time-consuming. The use of CrossValidator and ParamGridBuilder allowed for efficient hyperparameter tuning and model selection.

d. User Interface: Developing a user-friendly interface that allowed easy access to the news classification system was a challenge. By using web technologies like HTML, CSS, and JavaScript, we created a responsive and interactive interface, enhancing the user experience.

In conclusion, this project served as a valuable learning experience, exposing our team to various aspects of data processing, machine learning, and system design. While our system achieved an accuracy of 78%, we acknowledge that there is room for improvement. As we continue to refine our models and techniques, we aim to enhance the performance of our news classification system further.

6. Team Contributions

All team members played a vital role in the development and success of the project, with each member focusing on specific aspects to ensure a well-rounded and efficient big data system for news classification.

Tarun Reddy Nerella was responsible for researching and selecting suitable data sources and APIs for gathering news articles. They also developed the Kafka Producer and Consumer for handling real-time data streaming and were actively involved in discussions and decision-making processes regarding technology choices and system design.

Jagat Mohit led the data preprocessing tasks using NLP libraries like NLTK and spaCy. They collaborated with Team Member 3 in implementing the machine learning pipeline using PySpark and assisted in model evaluation and hyperparameter tuning. Additionally, they contributed to writing the project report, specifically the sections on data preprocessing and model evaluation.

Anudeep Kalitkar was focused on developing the machine learning pipeline using PySpark, which included feature extraction, model training, and model evaluation. They worked closely with Team Member 2 in selecting and implementing the classification models and implemented the MLOps concepts, such as model retraining and redeployment. They also assisted in developing the MongoDB storage solution.

Raghavendra Shanthamraju designed and implemented the MongoDB storage solution for storing classified news articles and contributed to integrating MongoDB into the overall architecture of the system. Furthermore, they developed the front-end web application or REST API for accessing the news classification system and participated in discussions and decision-making processes regarding technology choices and system design.

The collaborative effort of all team members ensured the successful development of a robust and efficient news classification system.