

Fake News Detection

Big Data Analytics - MS4

Amir ALI, Jacek CZUPYT, Javier SERRANO, Jean-Baptiste SOUBARAS

December 2022

Contents

1	Description of the subject	3
1.1	Problem statement	3
1.2	Introduction	3
1.3	Significance of the study	3
1.4	Non-functional objectives	4
2	Data sources	4
2.1	Streaming data	4
2.2	Batch data	4
3	Pre-processing and features extracted	5
3.1	Text Cleaning	5
3.2	Preprocess Operation	5
3.3	Feature Extraction	5
4	Technical Platform	6
4.1	System Architecture	6
4.2	Tools employed	7
5	Data ingestion and ETL	7
5.1	Twitter API	8
5.1.1	Testing	9
5.2	Reddit API	12
5.3	Batch labeled data	12
5.3.1	Creation of an artificial stream	13
5.3.2	ETL of the data	13
5.3.3	Storage in Hive	14
5.3.4	Tests	14
5.4	Streaming Layer	16
5.4.1	Tests	16

6	Machine Learning Implementation	16
6.1	Exploratory data analysis	16
6.2	Machine Learning Model	19
6.3	Result Evaluation	20
6.4	Model Versioning	21
7	Serving Layer	21
7.1	Individual Statement Query	22
7.2	Historical Data Query	22
7.3	Tests	22
	7.3.1 Conveying of tweets until the serving layer	22
	7.3.2 Historical Query testing	23
8	Business Perspectives and Ethical Issues	23
8.1	Business Perspectives	23
8.2	Ethical Issues	24

1 Description of the subject

1.1 Problem statement

We are sure you have heard about "Fake news" plenty of times recently. This is a topic that has gained exponential fame with the enormous growth of the Internet and especially, social networks over the last few years.

This outburst of social media flooded the internet with information, making it a competitor to traditional media. However, contrary to this latter, no accreditation nor proofreading process is required to post on social media. This means that everyone can post anything wanted, which makes it very easy to post fake news. In the end, it results in the spread of misinformation.

1.2 Introduction

The fake news on social media and various other media is wide spreading and is a matter of serious concern due to its ability to cause a lot of social and national damage with destructive impacts. The purpose of our project is to treat posts written on different social media in order to check whether a statement on the actuality is false or not. This will imply being able to stream data from social media sources, and train a machine learning-based model involving natural language processing methods to classify the different posts as "reliable" or "unreliable".

1.3 Significance of the study

This project could be useful in the context of fighting against misinformation and would allow automation of the fact-checking process which is today performed manually by journalists in order to inform social media users more effectively of the veracity of the information that can be stumbled upon while browsing the internet.

1.4 Non-functional objectives

The non-functional objectives of this project will be :

- efficiency, deploying a reliable classification model;
- maintainability, assuring the continuous functioning of the platform;
- ergonomics, creating an efficient front-end able to deal with a large range of queries;
- security, ensuring that the platform can't be deflected by someone willing to discredit real information or propagate fake news;
- ethics, ensuring that the errors made by the platform will not falsely harm the reputation of a journalist or politician or wrongly support and spread manipulated information.

2 Data sources

2.1 Streaming data

To process the news in real-time, we will have to stream posts sent on popular social media: Twitter, Reddit, and possibly Facebook. We will access the data using the APIs related to these networks.

The Twitter API allows us to stream or search for recent or historic tweets filtered by things such as keywords or hashtags. This should allow us to easily extract up-to-date news and events. It also allows us to receive up to 500k per month on its free tier, which should be more than sufficient for our purposes.

The Reddit API allows us to search for recent threads on a given subreddit, as well as stream updates on a particular thread. We plan to create a client which queries a single, or multiple news-related subreddits, and possibly converts them into a stream. The API allows for up to 60 calls/per minute.

The Facebook API appears to give the ability to read the feed of selected pages, letting us query various news pages for new posts. However preliminary research suggests that this may require special permissions from said pages that we may not be able to get access to.

2.2 Batch data

To ensure the classification results of our machine learning algorithms, we will need to train them on already classified data sets giving the reliability of different social media statements or articles. Some fact-checking organizations released such data sets containing short statements or titles of articles manually labeled as real or fake. These data sets will feed the batch layer of our architecture (detailed in the next section) and will be used as training data sets.

We will use three of these data sets:

- The *Fakeddit* dataset
- The *LIAR* dataset
- The ISOT dataset

The *Fakeddit* data set is detailed in the article [3]. It contains a classification of statements posted on the social media Reddit in 6 classes (True, Satire, Misleading, Manipulated, False Connection, and Imposter). The features used for this classification are the content of the different posts (text and image) and the comments. The data set (without taking into account the images that we won't use) is about 900 MB with the comments, and 200 MB with only the posts.

The *LIAR* data set consists of short statements made on the media (social or traditional) manually classified according to their degree of reliability (in a total of 6 classes: true, false, half-true, pants-fire, barely-true, mostly-true). The data set comes from the fact-checking American organization *PolitiFact* and was introduced in [4]. It contains 12.8K of statements, which make up about 3 MB of data.

The *ISOT* is a data set introduced in [1] and in [2], classifying articles from the press as true or fake. It contains about 45 K articles and is more than 100 MB.

The only information that will be exploited from these data sets are the statement/title and the binary classification as "fake" or "real" news (for the data sets which label in more than two classes, the closest class will be chosen).

3 Pre-processing and features extracted

To build a Model on text data first we need to preprocess a Text which is a very essential part of text classification applications.

3.1 Text Cleaning

To Clean the text first we convert the text into lowercase to give word the same preference. Afterward, we remove Punctuation, URLs, @tags, and Special Characters with the help of Regular Expressions. Because these are meaningless and are not so important for our application.

3.2 Preprocess Operation

After Cleaning the Text, first we build a pipeline for preprocessing text operation. This pipeline will be until the continuation of Building the Model and Prediction of the result based on the Stream data. But now we did until Feature Extraction. In the preprocess operation we make a token of each word and remove stopwords like "A", "THE", "IS", "AM" etc using `pyspark.ml.feature`.

3.3 Feature Extraction

Feature Extraction is a very essential part of Text Classification applications. In this part, we implement `CountVectorizer` and `IDF` using the `pyspark.ml.feature`. In this step first, we convert our text into numerical numbers based on semantic information of the Text and then transform it into `IDF` and took max feature based on `Wordcloud` information.

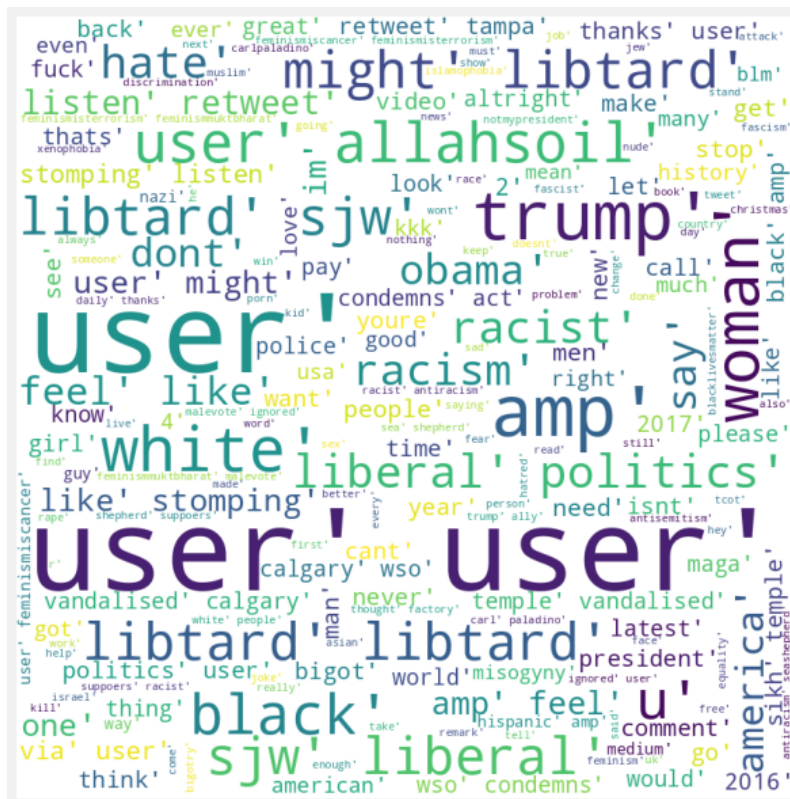


Figure 1: WordCloud of Fake News

4 Technical Platform

4.1 System Architecture

Figure 21: Shows our proposed architecture. It describes layers of the system and the technologies that we will use in each layer. The proposed architecture is based on Lambda which is divided into the batch, serving, and speed layers.

- **Batch Layer:** In this layer, we store all the raw data which come from Data Source and perform batch processing on the data. We plan to use Apache Hadoop for storage and Apache Spark for processing.
- **Machine Learning:** Fake news detection needs a text classification approach. To predict whether the given input is fake news or not we will use some machine learning classification methods.
- **Speed Layer:** This layer will analyze the real-time data. We plan to use Apache Spark or Flink for the processing.

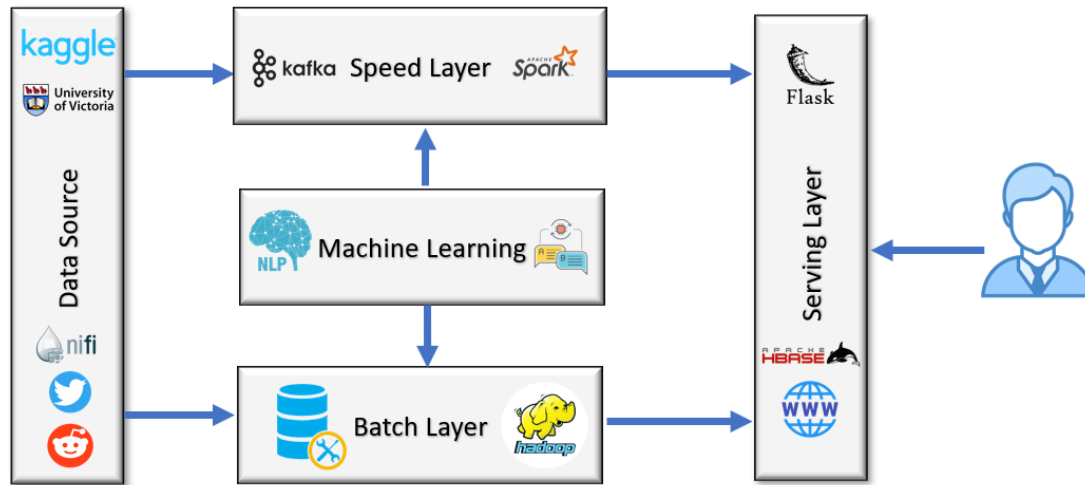


Figure 2: Our adoption of the lambda architecture

- **Serving Layer:** The serving layer will contain batch views and text data of fake news. We plan to use Apache Hbase for storage, Flask for the back-end server, and possibly some front-end framework such as React.

4.2 Tools employed

The platform was implemented on a private remote server. The software used were the following ones :

- Apache Hadoop 3.3.4;
- Apache Hive 3.1.2;
- Apache NiFi 1.18;
- Apache Kafka 2.6;
- Apache Spark 3.3.0.

The APIs of the social media *Twitter* and *Reddit* were used to collect the stream data. Finally, some coding was written in Python 3.7 using libraries *numpy*, *pandas* and *nlTK*.

5 Data ingestion and ETL

We use Apache Nifi to handle the ingestion and preliminary preprocessing of data. Its primary function is to manage the streaming data coming from the twitter and reddit apis, it can also handle preparing new labeled data, which can later be used to update our machine learning model.

5.1 Twitter API

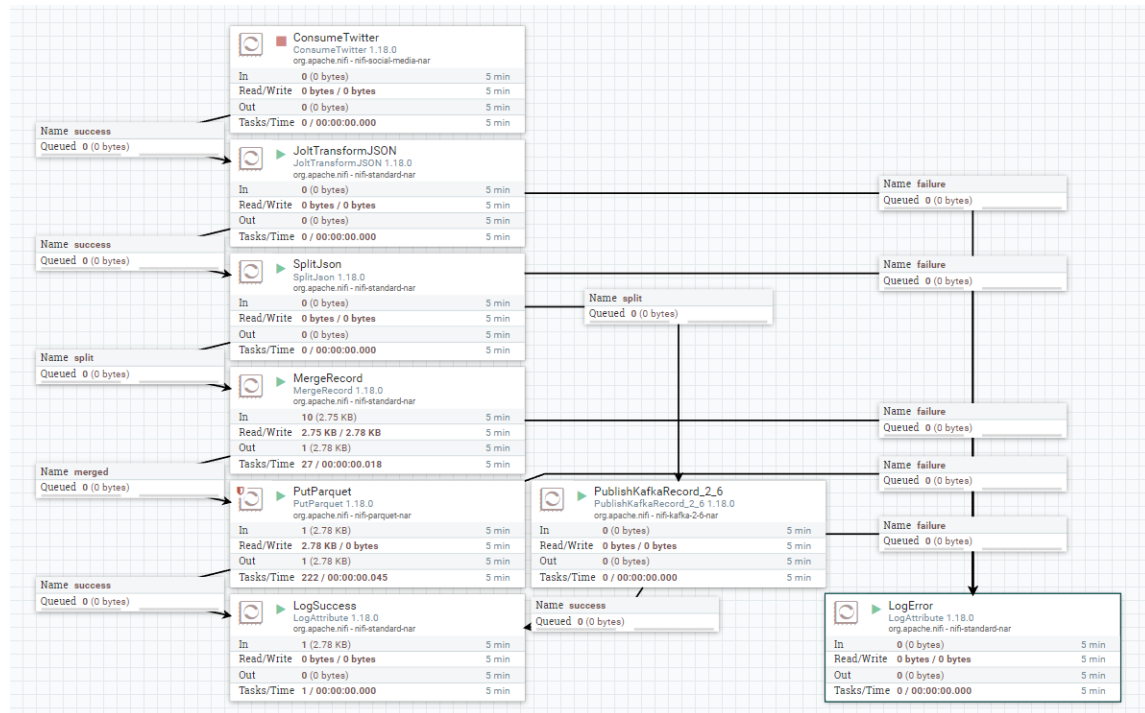


Figure 3: Twitter streaming api data flow

The Twitter steam ingestion dataflow (figure 3) is responsible for downloading live streaming data from the twitter api.

It begins with the inbuilt **ConsumeTwitter** processor, which is configured to use the Twitter v2 filtered streaming api. The exact filters are not sent together with the request for the stream, but instead preconfigured on a different endpoint. Given that our application has no need to automatically change filters, nor would this operation be in any way scalable, we decided against including this in the nifi flow. For the time being the filters have been manually configured only allow for english tweets with the hashtag "news", and we plan to provide scripts for changing the filters for the final milestone.

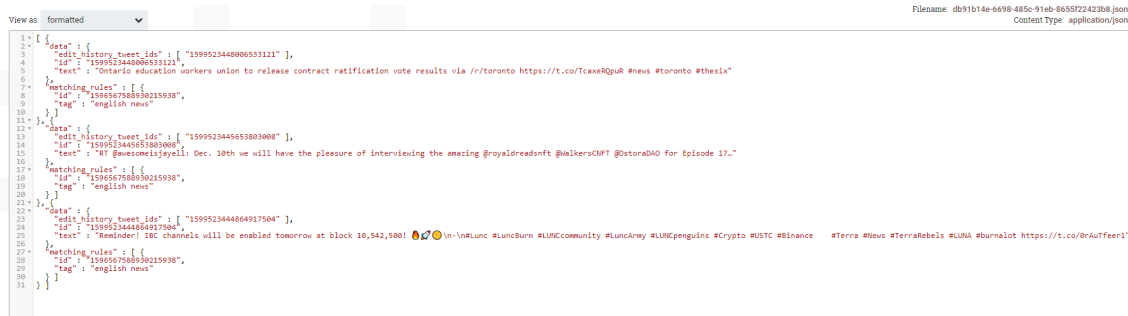
The tweets arrive in small groups, which are then stripped of redundant metadata about the stream filters, and split up into individual tweets. This data is then sent down two separate paths, firstly to a Kafka queue, from where it can then be consumed by the services in the streaming layer. Secondly, the individual tweets are grouped into sets of size up to 1000, and saved as parquet files, for use in the batch layer.

Where applicable (connections to external API / hadoop), retries with backoff are used. All other failures are logged along with they data they were attempting to process. Successful terminations are also logged, but without a copy of the data.

5.1.1 Testing

ConsumeTwitter processor test

After correctly configuring the ConsumeTwitter processor and connecting its output, it was turned on, and its queue was observed. As expected, the number of items in the queue started gradually rising. The contents of the output queue was then inspected (sample element in figure 4), which confirmed that the received tweets were written in english and contained the hashtag "news".



```

1 {
2   "data": {
3     "edit_history_tweet_ids": [ "1599523448006533121" ],
4     "id": "1599523448006533121",
5     "text": "Ontario education workers union to release contract ratification vote results via r/toronto https://t.co/TaxeRQGuR #news #thesix"
6   },
7   "matching_rules": [ {
8     "id": "1599567588930215938",
9     "tag": "english news"
10  } ],
11 },
12 {
13   "data": {
14     "edit_history_tweet_ids": [ "1599523448006533121" ],
15     "id": "1599523448006533121",
16     "text": "RT @auesome1joyell: Dec. 10th we will have the pleasure of interviewing the amazing @royaldreadnft @halkersCHFT @Dstonard40 for Episode 17..."
17   },
18   "matching_rules": [ {
19     "id": "1599567588930215938",
20     "tag": "english news"
21  } ],
22 },
23 {
24   "data": {
25     "edit_history_tweet_ids": [ "1599523448006533121" ],
26     "id": "1599523448006533121",
27     "text": "Reminder! IDC channels will be enabled tomorrow at block 10,542,500! 🐧🐧🐧\n\n#Lunc #LuncBurn #LuncCommunity #LuncArmy #Luncpenguins #Crypto #USTC #Binance #terra #news #TerraRebels #LUNA #burnalot https://t.co/BrAulfeer1"
28   },
29   "matching_rules": [ {
30     "id": "1599567588930215938",
31     "tag": "english news"
32  } ],
33 }

```

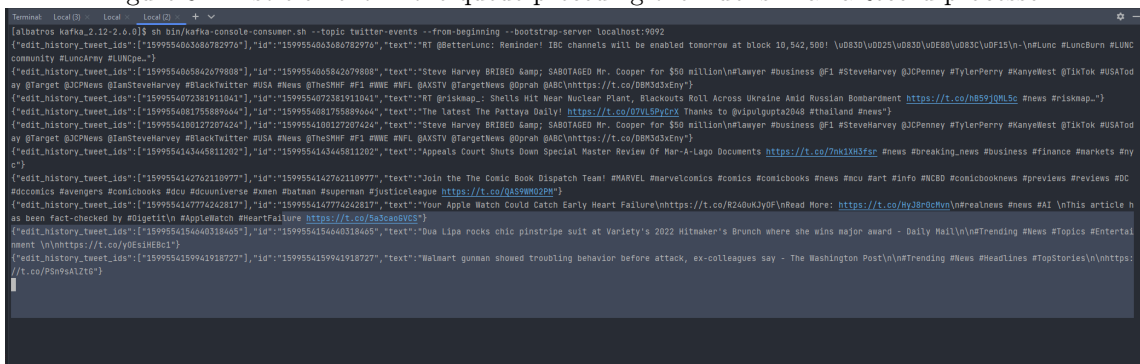
Figure 4: Sample element in the ConsumeTwitter output queue

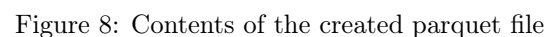
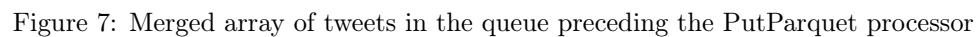
Twitter to Kafka test

In order to verify that flow file correctly transfers the data from the Twitter stream to the Kafka queue, the individual processors were turned on in sequence, ensuring that no errors occur, and the data in the given queue is in the expected format. After all processors have finished, the Kafka host was accessed directly via terminal to verify that the tweets have indeed been correctly published. The figures 5 and 6 show the same tweet, first in the queue preceding the PublishKafkaRecord processor, and later inside the designated Kafka topic.

Twitter to Parquet test

The flow of data from the Twitter stream to the parquet files was then verified in a similar manner. For convenience, during the test, the minimum number of elements for the MergeRecord processor was lowered to 10. After checking the integrity of the data in between each processor, the new file has indeed been created in the designated location. It was then downloaded and its content was examined using and python "pandas" library. The figures 7 and 8 shows that the contents of the parquet file matches the tweets previously seen in the nifi queue.





5.2 Reddit API

We used Apache nifi to extract and use the Reddit API data. We started by putting two processors, InvokeHTTP and PutFile in nifi, and connecting them from InvokeHTTP to PutFile. After that, it was needed to specify some things of their configuration. We needed to access the Reddit API and we got it in a .json. Then, we specified the URL to the InvokeHTTP processor, and it was also needed to export its certificate for what we used Java keystore. Finally specify other things of the configuration of the processors.

When trying to access the Reddit API we faced some problems. First of all, we needed to fill a form specifying a lot of personal data, and explaining why we wanted to access the API, and even doing that, we did not get access to the API so we had to try it by another way. Other problems faced were that we needed to import the certificate of the API via Java Keystore which I did not have installed in my computer, so I tried to install it but my computer did not let me do it. Finally, we also faced some problems when specifying the SSL context service, when clicking it did not open the corresponding window but I think that was some kind of bug in the app.

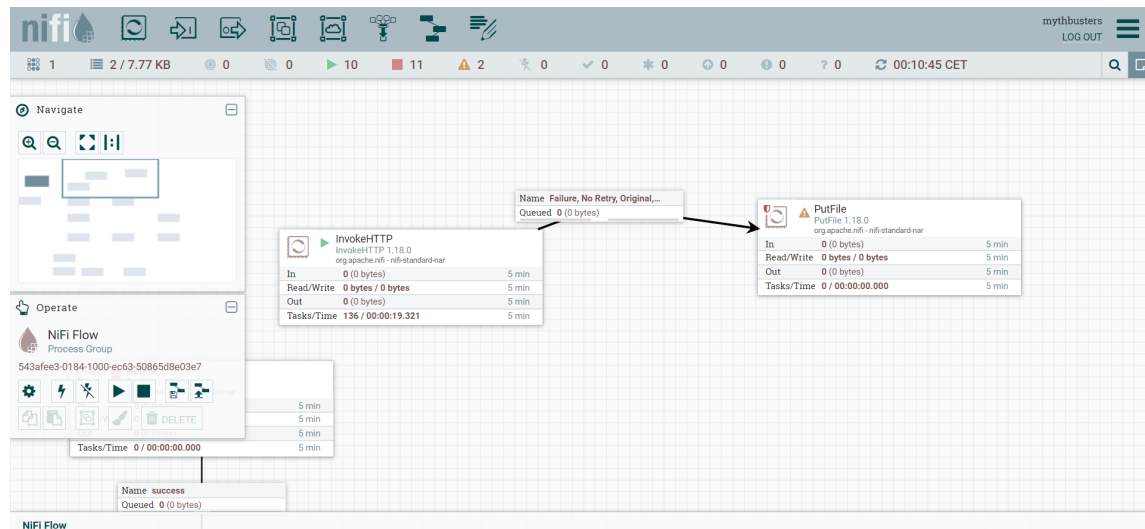


Figure 9: Processors in nifi to get the data from Reddit API

5.3 Batch labeled data

The ingestion of the labeled data comes in three steps:

1. Conversion of the data from batch to stream;
2. Conversion to *parquet* format on HDFS;
3. Insertion of the data into a Hive table.

5.3.1 Creation of an artificial stream

One of the requirements of the projects is to be able to regularly update the analysis ML model by training it on new data. However, this project needs labeled data for supervised learning, and the existing data sources for labeled fake news detection are only in batch format. It is thus necessary to convert this data into a stream. To do so, the model is initially trained on only a part of the data. Then, at a given rate, new labeled instances are loaded into the platform at a given rate, and are regularly merged, transformed, and loaded into Hive, in order to eventually update the model.

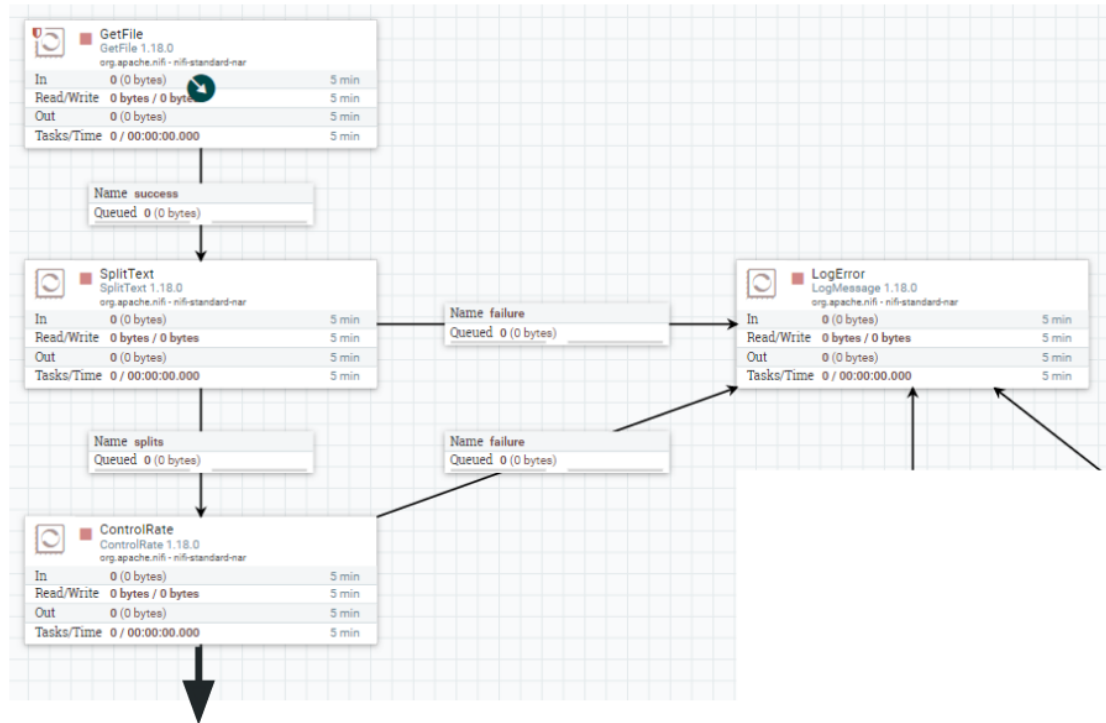


Figure 10: Creation of an artificial stream from the batch data.

The new data are merged and loaded every day, partitioned according to their date of emission. In a real business application, instead of a set of batch data, the best solution would be to implement several web scrapers able to collect information on given fact-checking sources such as websites or social media account of fact-checking organizations that regularly upload new content. This exceeds the scope of this work, which is focused on learning purposes, that's why an artificial streamline is created from batch data instead.

5.3.2 ETL of the data

Once the data is streamed line by line, it needs to be regularly merged, then attributed a partition name, and finally converted to parquet in the corresponding partition.

There is an implicit buffering step before the merging of the file, since that operations is only

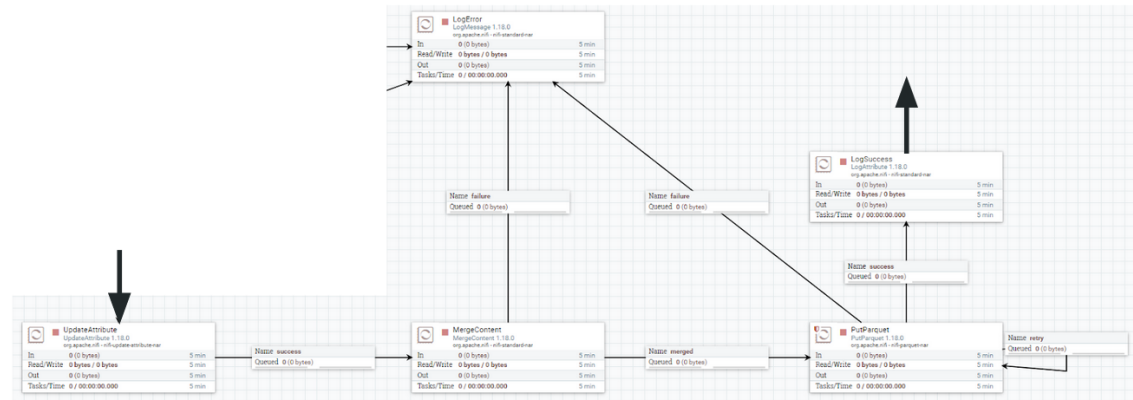


Figure 11: Conversion of the data into parquet and loading into HDFS.

executed once every day, but the control of the rate of the stream makes it not very useful to use external buffering tools (such as Kafka) since the size of the buffer can be easily maintained low.

5.3.3 Storage in Hive

To store the data in Hive, an external partitioned table is created in parquet format following the schema of the input data. The location of this table needs to be that of the parquet files loaded at the previous step. Then to update the Hive table, only the metadata needs to be updated, which is done via the hive command `MSCK REPAIR TABLE`.

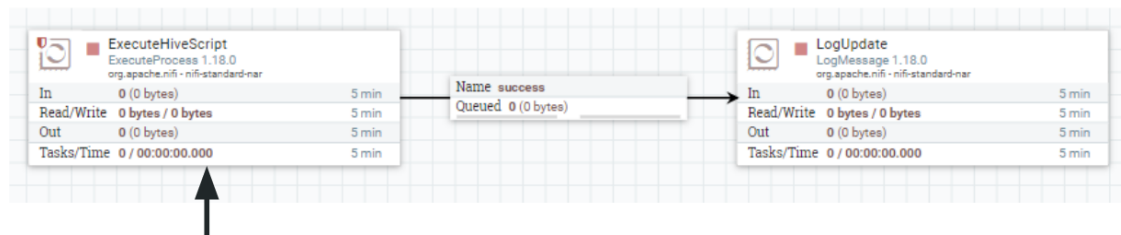


Figure 12: Updating the Hive table.

5.3.4 Tests

This flow line was tested with a set of mock data for iris classification. The data set has 150 rows. The figures 13, 14 show the content of the Hive table during the process and at the end.

The data contained in the table has not been altered, and the number of rows in the table corresponds to the number of instances received. This asserts the quality and completeness of the process.

```

Logging initialized using configuration in jar:file:/users/elevs-b/2018/jean-ba
ptiste.soubaras/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.pro
perties Async: true
Hive Session ID = f52c2548-68d7-4158-85f5-a816efc973ba
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive (default)> use mythdb;
OK
Time taken: 0.317 seconds
hive (mythdb)> select * from flwr_parquet;
OK
5.5      2.4      3.8      1.1      Versicolor
5.5      2.4      3.7      1.0      Versicolor
5.8      2.7      3.9      1.2      Versicolor
6.0      2.7      5.1      1.6      Versicolor
5.4      3.0      4.5      1.5      Versicolor
6.0      3.4      4.5      1.6      Versicolor
6.7      3.1      4.7      1.5      Versicolor
6.3      2.3      4.4      1.3      Versicolor
5.6      3.0      4.1      1.3      Versicolor
5.5      2.5      4.0      1.3      Versicolor
Time taken: 1.466 seconds, Fetched: 10 row(s)
hive (mythdb)>

```

Figure 13: After having ingested the first 10 entries.

```

5.1      3.9      1.4      0.2      Setosa
4.9      3.0      1.4      0.2      Setosa
4.7      3.2      1.3      0.2      Setosa
4.6      3.1      1.5      0.2      Setosa
5.0      3.6      1.4      0.2      Setosa
4.4      3.9      1.7      0.4      Setosa
4.6      3.4      1.4      0.3      Setosa
5.0      3.4      1.5      0.2      Setosa
4.4      2.9      1.4      0.2      Setosa
4.9      3.1      1.5      0.1      Setosa
5.4      3.4      1.7      0.2      Setosa
5.1      3.7      1.5      0.4      Setosa
4.6      3.6      1.0      0.2      Setosa
5.1      3.3      1.7      0.5      Setosa
4.8      3.4      1.9      0.2      Setosa
5.0      3.0      1.6      0.2      Setosa
5.0      3.4      1.6      0.4      Setosa
5.2      3.5      1.5      0.2      Setosa
5.2      3.4      1.4      0.2      Setosa
4.7      3.2      1.4      0.2      Setosa
7.0      3.2      4.7      1.4      Versicolor
6.4      3.2      4.5      1.5      Versicolor
6.9      3.1      4.9      1.5      Versicolor
5.5      2.3      4.0      1.3      Versicolor
6.5      2.8      4.6      1.5      Versicolor
5.7      2.8      4.5      1.3      Versicolor
6.3      3.3      4.7      1.6      Versicolor
4.9      2.4      3.3      1.9      Versicolor
6.6      2.9      4.6      1.3      Versicolor
5.2      2.7      3.9      1.4      Versicolor
Time taken: 1.445 seconds, Fetched: 150 row(s)
hive (mythdb)>

```

Figure 14: The final content of the table.

5.4 Streaming Layer

In the streaming layer, we utilize Apache Kafka in order to buffer the streams of data arriving from the Nifi services in the ingestion layer. From there we use Apache Spark, using the pyspark library, in order to efficiently process the data and send the results to the serving layer.

The pyspark application loads the created machine learning model into memory upon being initialized. In the case where the model is updated, the spark process must be restarted in order for the model to be reloaded.

The spark application is subscribed to the relevant Kafka topics, which hold the buffered data, and pass any new data through the model. It then writes the results into a parquet file, where it can be loaded by the serving layer.

The relevant code is contained in the *spark.py* python file, starting method in the *launch_pyspark.sh* bash server script.

5.4.1 Tests

The previously described Nifi flow was activated to gather sample tweets, and place them into the Kafka topic. The spark process was running, and received the new data from Kafka, evaluated it with the model, and sent it to the serving layer. Figures 15 and 16 show the same news tweet present in first the Kafka buffer, and then evaluated in the serving layer.

```
{
  "edit_history_tweet_ids": [
    "1617555064960475137"
  ],
  "id": "1617555064960475137",
  "text": "France pension protests: Man loses testicle after being clubbed by police #FranceNews #FrenchNews #FrenchPolitics #EuropeNews #News [Video] https://t.co/rJMo3tLSDR"
}
```

Figure 15: Tweet present in the Kafka buffer

```
France pension protests: Man loses testicle after being clubbed by police #FranceNews #FrenchNews #FrenchPolitics #EuropeNews #News [Video] https://t.co/rJMo3tLSDR
2023-01-23 17:09:21.005 1
```

Figure 16: Tweet from figure 15 now present in serving layer

6 Machine Learning Implementation

6.1 Exploratory data analysis

The figures 17 and 18 show the comment's character count and word count distributions, respectively. Most comments are rather short, averaging around 600 words or 4000 characters. There are no zero-length comments, as those are impossible to post on the site, but there are a few single-character comments. The longest comment has 2000 words and 15000 characters.

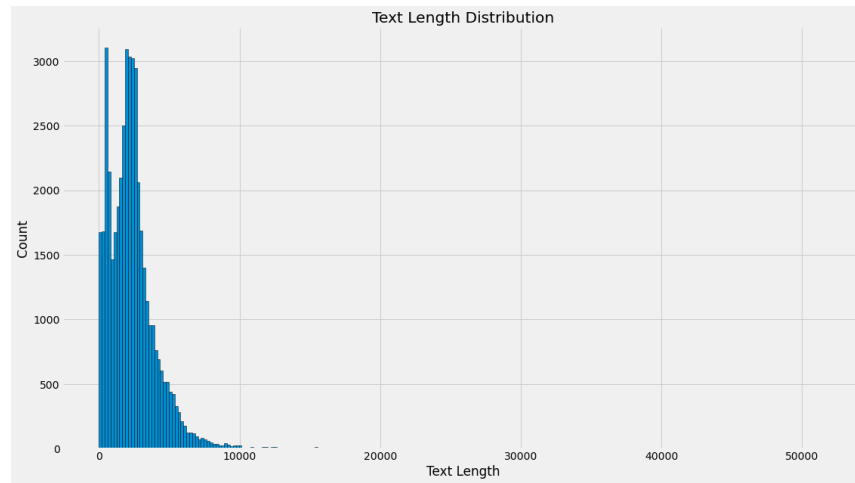


Figure 17: Comment Character Count Distribution

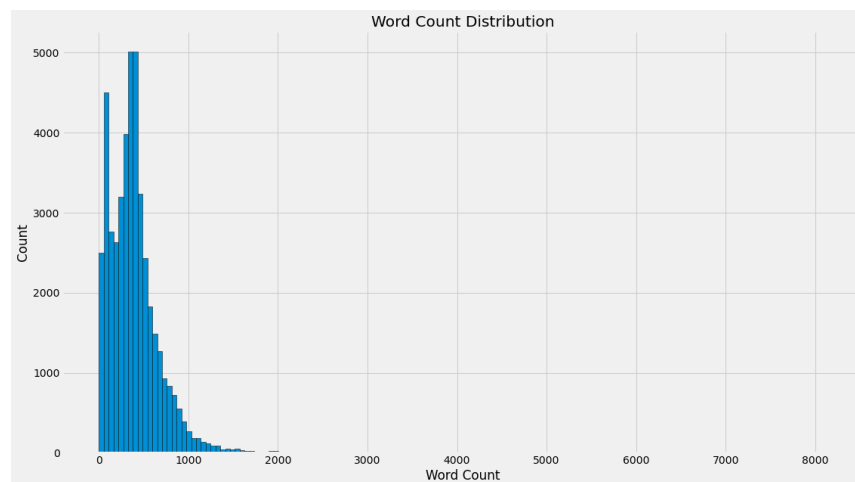


Figure 18: Comment Word Count Distribution

It's very important to understand the context of the text, especially when working on Text Extraction because every word's meaning is important. In order to better understand the sentiment of the text; we used the TextBlob library to find the estimated sentiment polarity of the comments. The results are present in figure 19. The polarity score ranges between -1 and 1, where positive values represent a positive sentiment and negative values represent a negative sentiment. The graph shows that the sentiment are also equally distributed.

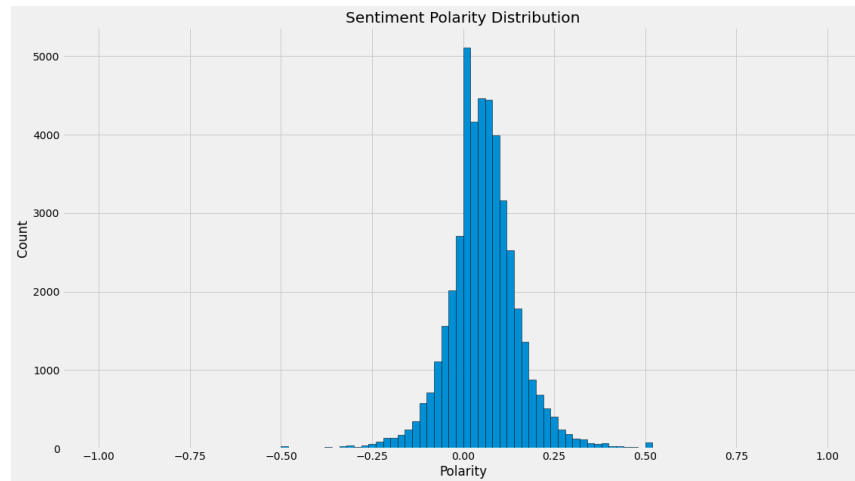


Figure 19: Review Rating Distribution

Based on the historical data, the distribution of fake news, and true news is almost equally distributed, and there is no in-balance issue below; you can see the pie chart.

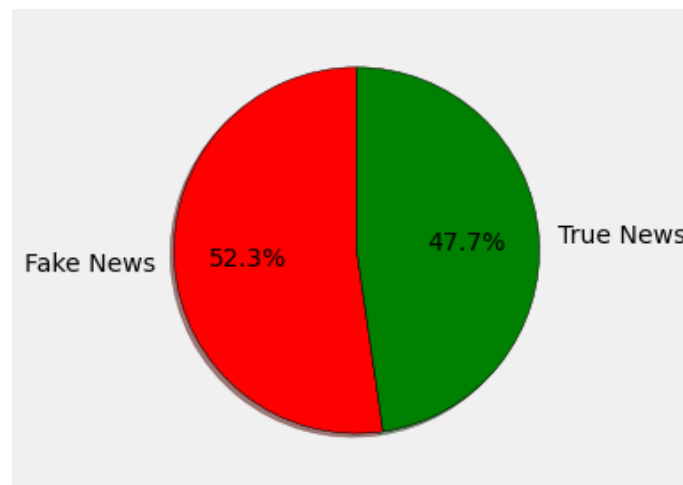


Figure 20: Distribution of Label Class

A word cloud is a visual representation of the frequency of words within a given text. It is typically created by displaying a list of words, with the size of each word representing its frequency or importance within the text. The most common words are usually displayed in larger fonts, while less common or less important words are displayed in smaller fonts. Our historical data is mostly based on news of the USA so below you can see most larger words with American politician's information.

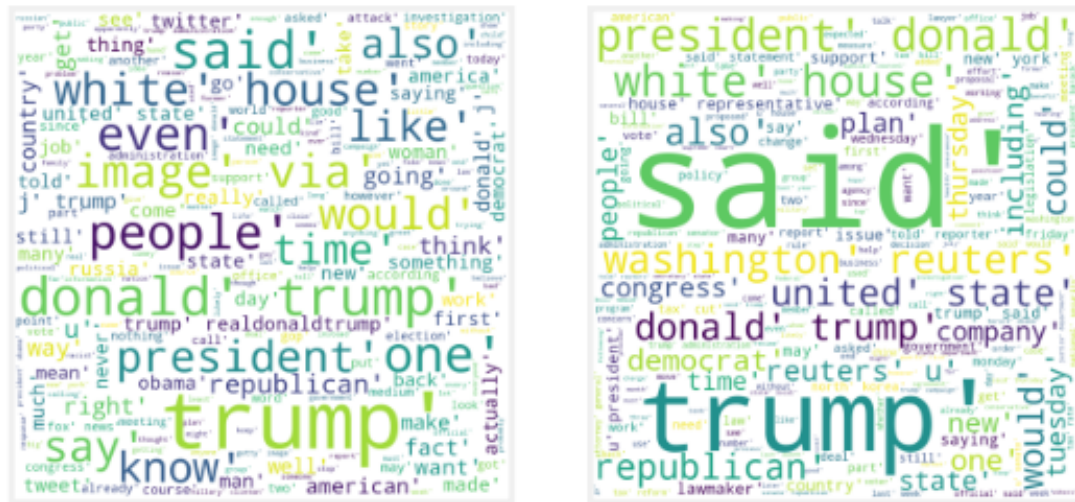


Figure 21: WordCloud of Fake vs true News

6.2 Machine Learning Model

Several machine learning algorithms can be used to detect fake news, including logistic regression, Naive Bayes, decision trees, random forests, K-nearest neighbors (KNN), support vector machines (SVM), and multilayer perceptrons (MLPs).

Logistic Regression: This is a simple yet powerful linear model that can be used for binary classification tasks. It is often used as a baseline model because it is easy to implement and interpret.

Support Vector Machines (SVMs): These are powerful algorithms that can be used for both classification and regression tasks. They work by finding the hyperplane in a high-dimensional space that maximally separates different classes.

Decision Trees: These are tree-like models that make predictions based on a series of binary splits. They are simple to understand and interpret, but can be prone to overfitting if not properly tuned.

Random Forests: These are ensemble models that combine the predictions of multiple decision trees. They tend to be more accurate than individual decision trees, and are resistant to overfitting.

Neural Networks: These are complex models that are inspired by the structure and function of the human brain. They can be used for a wide range of tasks, including classification, and can achieve state-of-the-art results on many problems.

In Apache Spark, we use these algorithms to detect fake news by following these steps:

- Preprocess the text data by tokenizing it, removing stop words, and performing other necessary cleaning tasks.
- Vectorize the text data using a feature extractor such as CountVectorizer or TfidfVectorizer. This converts the text data into numerical feature vectors that can be used as input to the machine learning algorithms.
- Split the preprocessed and vectorized data into training and test sets.
- Train a logistic regression, Naive Bayes, decision tree, random forest, KNN, SVM, or MLP model on the training data.
- Evaluate the model on the test data and calculate its performance metrics, such as accuracy, precision, and recall.

Here is an example of how we use logistic regression for fake news detection in Apache Spark:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import CountVectorizer

# Preprocess and vectorize the text data
vectorizer = CountVectorizer(inputCol="text", outputCol="features")
data = vectorizer.fit(df).transform(df)

# Split the data into training and test sets
(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Train a logistic regression model
lr = LogisticRegression(maxIter=10, regParam=0.01)
lrModel = lr.fit(trainingData)
```

Figure 22: Implementation of Pyspark in Logistic Regression

To use Naive Bayes, decision trees, random forests, KNN, SVM, or MLP for fake news detection in Apache Spark, we follow a similar process, but use the appropriate classifier from the `pyspark.ml.classification` module and set necessary hyperparameters according to the model requirement.

6.3 Result Evaluation

Ultimately, the choice of which machine learning model to use for fake news detection will depend on the specific requirements of the task, the availability of data and resources, and the desired level

of accuracy. In this project, we implement different machine learning models, compare the result, and choose the best model for the application. Below you can see the comparison result; clearly, we can see that Logistic Regression gave much better results than others.

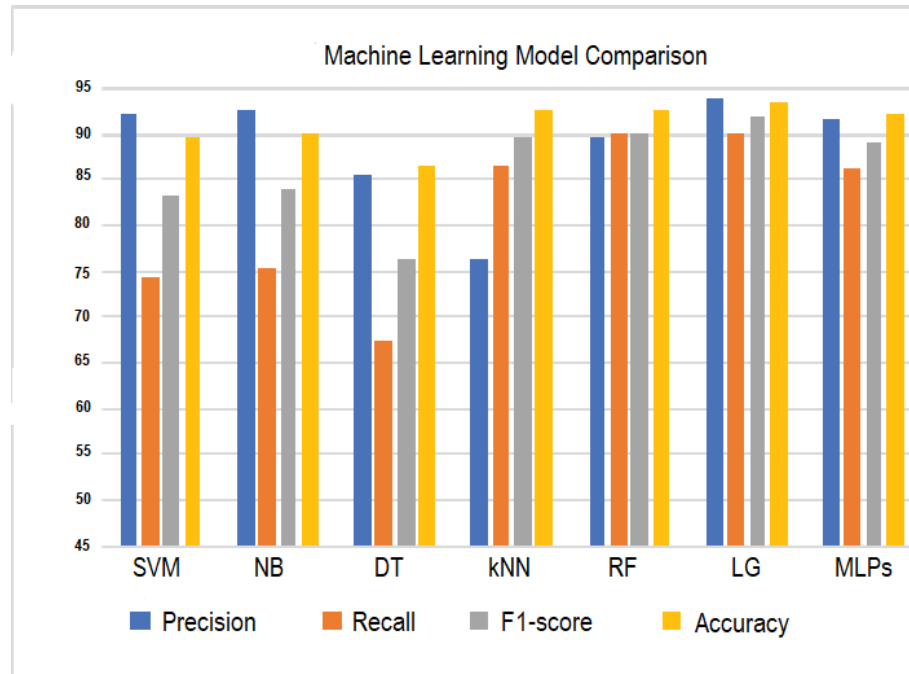


Figure 23: Result Evaluation

6.4 Model Versioning

To take into account the increasing amount of labeled data arriving through the batch layer on Hive, the model is re-trained on a regular basis on the data of the table. To allow the speed layer to use the latest version of the model, and to be security resilient, all the successive versions of the model are stored in the local memory of the server. On a larger scale system (such as a cluster), it would be best to consider several duplicate copies of the model.

7 Serving Layer

The serving layer was created to allow an user to make two types of queries:

- Individual statement query
- Historical Data query.

The first type consists in simply entering a statement and getting the model prediction of its veracity, which requires no other data than the trained model. The second type of data require to

store aggregated historical data in a table able to answer queries such as: How many fake news have been posted today ? What is the proportion of fake news among all news in the last week ?

7.1 Individual Statement Query

The trained model has been integrated into a Flask application. It allows the user to enter the statement to predict and the application will run the model on it and return FAKE or TRUE.



Figure 24: Application Design

7.2 Historical Data Query

To be able to answer historical data based queries, processed stream data is transmitted through Kafka into a Hive table, keeping in memory each tweet, its timestamp and its prediction. Cleaning scripts can be executed on a regular basis to get rid of old data in order to limit the size of the table.

7.3 Tests

7.3.1 Conveying of tweets until the serving layer

To test the integrity of the data received from the API until the serving layer, and the correct execution of the stream processing, here is the example of a tweet published through Kafka (before processing), and the same tweet stored in the serving layer table (after processing), with a prediction label (1 means fake news, 0 means real news).

```
{ "id": "127955064908479137", "text": "France pension protests: Man loses testicle after being clubbed by police #FranceNews #FrenchPolitics #EU", "prediction": 1 }
```

Figure 25: Tweet published through Kafka.

```
France pension protests: Man loses testicle after being clubbed by police #Franc
eNews #FrenchNews #FrenchPolitics #EuropeNews #News [Video] https://t.co/rJMo3tL
SDR      2023-01-23 17:09:21.005 1
```

Figure 26: The tweet was stored in the batch views table, labeled as fake news.

7.3.2 Historical Query testing

The following query consists in getting the percentage of fake news among posts made in English under the hashtag #news for the last hour.

```
SELECT 100*SUM(last_pred.prediction)/COUNT(*)
FROM (
  SELECT prediction
  FROM batch_view
  WHERE (UNIX_TIMESTAMP(current_timestamp()) - UNIX_TIMESTAMP(datetimestamp) < 3600)
) AS last_pred;
```

Figure 27: Query executed.

```
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-01-23 15:22:53,706 Stage-1 map = 0%, reduce = 0%
2023-01-23 15:22:56,747 Stage-1 map = 100%, reduce = 100%
Ended Job = job_local1314716597_0006
MapReduce Jobs Launched:
Stage-Stage-1: HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
49.63503649635037
Time taken: 4.334 seconds, Fetched: 1 row(s)
hive (mythdb)>
```

Figure 28: Result of the query: 49.6% of the tweets posted in the last hour were fake news.

8 Business Perspectives and Ethical Issues

8.1 Business Perspectives

Implementing such a platform for business perspectives could be strategic for social media platforms, which are the more impacted by the spread of fake news. A social media platform with an efficient fake news detection system could gain an advantage on its competitors by offering its users tools to filter out misinformation and may avoid potential political decision that, in the future, would penalize the platforms who spread the most fake news. Some platforms, like Facebook, try to implement such systems, but on low scales (and with mitigated results).

To improve the solution presented in this project in the purpose of scaling it to business operations, four axis should be dealt with:

- Create a streaming source of recent labeled data (via web scraping on fact-checkers website for example);
- Removing the limitations provided by the free versions of the streaming APIs;
- Implementing the platform on a cluster to improve computer power and storage;
- Design an ergonomic and user-friendly front-end able to answer to detailed business-specific queries on top of the serving layer.

8.2 Ethical Issues

In spite of the opportunities that the normalization of such systems could bring, there are some ethical concerns that should be thoroughly examined before considering so.

First, any error of classification could have considerable impact:

- A fake news classified as real would spread much more easily than if it wasn't processed;
- On the opposite, a real news classified as fake would at the same time encourage the propagation of other contradicting fake news, and impair the reputation of the newspaper/journalist/celebrity responsible for the statement.

Moreover, the range of use of such methods should be clearly defined by the platforms using them. For example, wrongly using fake news detection algorithms on statements made by a comedian making fun of the actuality would first mislead the public into thinking he is not making jokes but spreading misinformation, and would potentially lead to punishing treatments on part of the social media platform (ban, demonetization, removal from the recommendation algorithms...).

Finally, one of the biggest concerns that can be expressed about this system is that if a black-box algorithm is used to declare which news are fake or real, and the cases of use are decided by the decision board of the social media without being made public, the platform can alter the functioning of the detection system to target the news that would go counter the business interests of the company, thus resulting in manipulation of the public opinion.

References

- [1] Saad S Ahmed H, Traore I. " detection of online fake news using ngram analysis and machine learning techniques. ", 2017.
- [2] Saad S Ahmed H, Traore I. Detecting opinion spams and fake news using text classification. *Journal of Security and Privacy*, 2018.
- [3] Kai Nakamura, Sharon Levy, and William Yang Wang. r/fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection. *arXiv preprint arXiv:1911.03854*, 2019.
- [4] William Yang Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.