



Université Chouaib Doukkali
Ecole Nationale des Sciences Appliquées d'El
Jadida



MODULE :

**ARCHITECTURE ET
INFRASTRUCTURE BIG
DATA**

Filière : **2ITE3**

**Projet 10 :
Mise en place d'un data lake pour la visualisation
des données financières en utilisant Apache Kafka,
Apache Spark, Apache Beam,
Apache Druid et Streamlit**

Réalisé Par :
AMINE ELMANSOURI
CHARAFEDDINE ELBAHJA
YASSINE HASSINI

Professeur :
Pr. FAHD KALLOUBI

ANNEE UNIVERSITAIRE : 2023-2024

Table des matières

1.	Objectif	5
2.	Installation du docker et Docker compose.....	5
1.	Installation de Docker :	5
2.	Vérification de l'installation de Docker :.....	5
3.	Installation de Docker Compose :	5
4.	Vérification de l'installation de Docker Compose :	5
3.	Configuration de l'Environnement Dockerisé.....	6
1.	Création des Conteneurs.....	6
2.	Exécution des conteneurs	9
4.	Configuration des api New York Times et Yahoo Finance	11
1.	Api New York Times	11
2.	Yahoo finance Api	15
5.	Interception des données des 2 Apis en utilisant Apache Beam	16
6.	Enregistrement des données brutes sous format Parquet.....	19
7.	Entraînement de plusieurs modèle ML pour sentiment analysis	19
8.	La lecture des données enregistrées avec Spark SQL.....	23
9.	Prédiction du score de sentiment des textes en utilisant Spark ML.....	25
10.	Ajout de nouvelles colonnes dans les données NYT	26
11.	Enregistrement du nouveau DataFrame obtenu dans apache Druid.....	29
12.	Visualisation du résultat de traitement effectue en utilisant Streamlit	36

Liste des Figures

Figure 1:Le contenu de fichier docker-compose.yml-partie1	6
Figure 2:Le contenu de fichier docker-compose.yml-partie2	6
Figure 3:Le contenu de fichier docker-compose.yml-partie3	7
Figure 4:Le contenu de fichier docker-compose.yml-partie4	7
Figure 5:Le contenu de fichier docker-compose.yml-partie5	8
Figure 6:Le contenu de fichier docker-compose.yml-partie6	8
Figure 7:Les volumes des conteneurs	9
Figure 8:Exécution des conteneurs	9
Figure 9:interface de docker Desktop	10
Figure 10:Interface des apis New York Times	11
Figure 11:Apis New York Times	13
Figure 12:Archive Api pour les articles de New York Times	13
Figure 13:La création d'une nouvelle app	15
Figure 14:Interception des données-partie1	16
Figure 15:Interception des données-partie2	17
Figure 16:Interception des données-partie3	18
Figure 17:Interception des données-partie4	18
Figure 18:Les fichiers parquet générés	19
Figure 19:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie1	20
Figure 20:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie2	20
Figure 21:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie3	21
Figure 22:Entraînement et évaluation des modèles	21
Figure 23:Enregistrement du meilleur modèle sur HDFS	22
Figure 24:Utilisation du modèle pour faire des prédictions sur de nouvelles données	23
Figure 25:Affichage des résultats des prédictions	23
Figure 26:La lecture des données enregistrées avec Spark SQL	24
Figure 27:L'affichage des résultats des requêtes SQL-partie1	25
Figure 28:L'affichage des résultats des requêtes SQL-partie2	25
Figure 29:Prédiction du score de sentiment des textes en utilisant Spark ML ...	26
Figure 30:L'ajout de nouvelles colonnes dans les données NYT-partie1	28
Figure 31:L'ajout de nouvelles colonnes dans les données NYT-partie2	29
Figure 32:L'interface utilisateur de Apache Druid	30
Figure 33:Le choix de Batch-classic	30
Figure 34:cliquer sur HDFS	31

Figure 35:Cliquer sur Connect data	31
Figure 36:L'insertion du chemin hdfs de nouveau dataframe.....	32
Figure 37:Le schéma de nouveau dataframe.....	34
Figure 38:L'ajout de nouveau dataframe est entrain de s'exécuter	34
Figure 39:L'ajout de nouveau dataframe est entrain est exécuté avec succès	35
Figure 40:La création de nouveau datasource druid "nouveau_dataframe"	35
Figure 41:Le tableau des données du nouveau_dataframe	38
Figure 42:Le nombre de sentiment positifs par jour	38
Figure 43:Le nombre de sentiment négatifs par jour	39
Figure 44:La moyenne des sentiments par jour	40
Figure 45:Le nombre des articles par jour	40
Figure 46:Répartition des scores de sentiments	41
Figure 47:Le nombre d'occurrences pour chaque section.....	41
Figure 48:Le graphe à barres pour calculer le nombre d'occurrences pour chaque section.....	42
Figure 49:Le résumé de nuage de mots.....	42

1. Objectif

Ce projet vise à établir un data lake pour visualiser des données financières en utilisant Apache Spark, Apache Beam, Apache Druid, et Streamlit. Deux sources de données, dont Yahoo Finance et l'API New York Times, sont intégrées via Apache Beam et enregistrées au format Parquet. En utilisant Spark ML, plusieurs modèles de sentiment analysis sont entraînés sur un ensemble de données de Kaggle, et le meilleur modèle est retenu. Les données sont ensuite analysées avec Spark SQL pour prédire les scores de sentiment NYT, et le résultat est enregistré dans Apache Druid. Enfin, Streamlit offre une interface interactive pour explorer les résultats, incluant le tableau de données et les statistiques de sentiment par jour etc...

2. Installation du docker et Docker compose

L'installation de Docker et Docker Compose se fait en plusieurs étapes. Voici une brève description des étapes pour l'installation de Docker et Docker Compose :

1. Installation de Docker :

- Rendez-vous sur le site officiel de Docker pour télécharger la version appropriée de Docker pour votre système d'exploitation : <https://docs.docker.com/get-docker/>
- Suivez les instructions d'installation spécifiques à votre système d'exploitation.

2. Vérification de l'installation de Docker :

- Après l'installation, exécutez la commande suivante dans un terminal ou une invite de commandes pour vérifier que Docker est correctement installé :

⇒ **docker --version**

3. Installation de Docker Compose :

- Docker Compose est généralement inclus avec Docker sur la plupart des plateformes. Si ce n'est pas le cas, suivez les instructions spécifiques à votre système d'exploitation pour installer Docker Compose : <https://docs.docker.com/compose/install/>

4. Vérification de l'installation de Docker Compose :

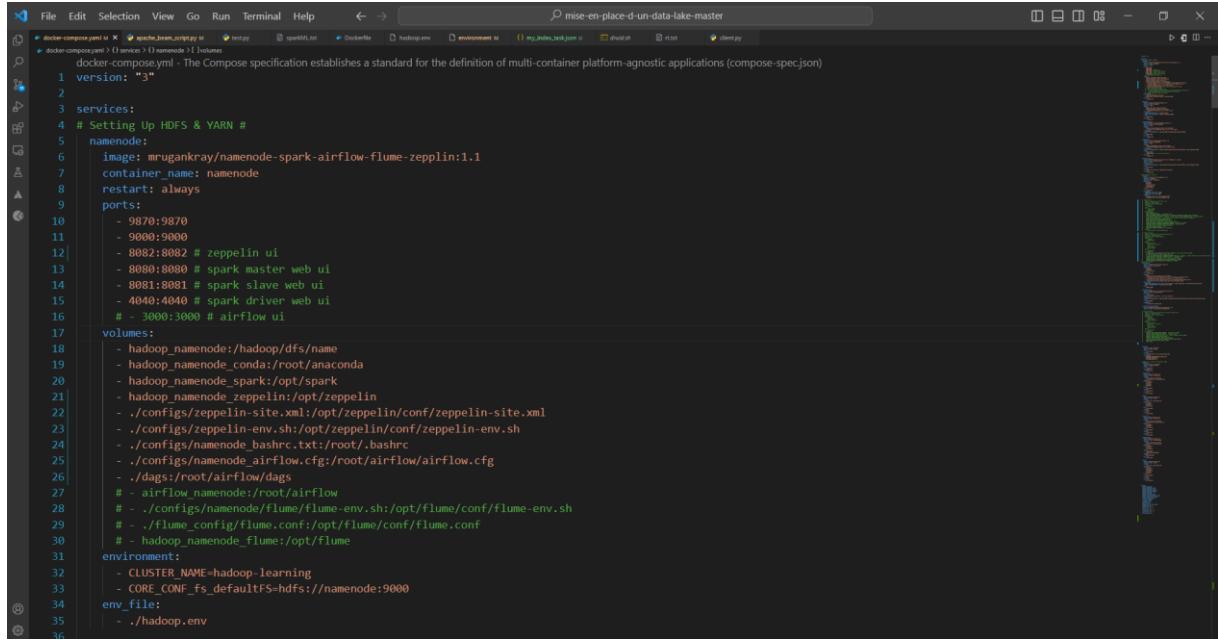
- Après l'installation, exécutez la commande suivante pour vérifier que Docker Compose est correctement installé :

⇒ **docker-compose –version**

3. Configuration de l'Environnement Dockerisé

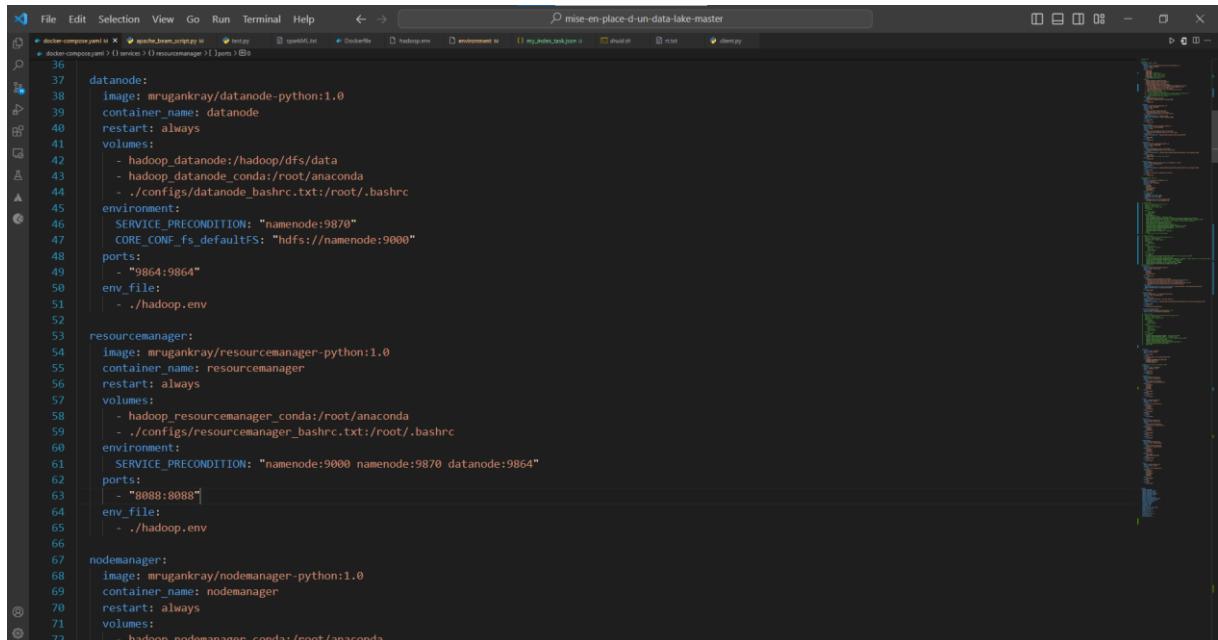
1. Crédit des Conteneurs

Élaborez un fichier docker-compose.yml décrivant les services nécessaires pour cette application se référer au fichier docker-compose.yml dans le dossier du projet.



```
File Edit Selection View Go Run Terminal Help < -> mise-en-place-d-un-data-lake-master
docker-compose.yml docker-compose spec.json Dockerfile logs environment.yml my_docker_link.json restart client.py
1 version: "3"
2
3 services:
4 # Setting Up HDFS & YARN #
5 namenode:
6   image: mrugankray/namenode-spark-airflow-flume-zeppelin:1.1
7   container_name: namenode
8   restart: always
9   ports:
10    - 9870:9870
11    - 9000:9000
12    - 8882:8882 # zeppelin ui
13    - 8080:8080 # spark master web ui
14    - 8081:8081 # spark slave web ui
15    - 4040:4040 # spark driver web ui
16    # - 3000:3000 # airflow ui
17   volumes:
18    - hadoop_namenode:/hadoop/dfs/name
19    - hadoop_namenode_conda:/root/anaconda
20    - hadoop_namenode_spark:/opt/spark
21    - hadoop_namenode_zeppelin:/opt/zeppelin
22    - ./configs/zeppelin-site.xml:/opt/zeppelin/conf/zeppelin-site.xml
23    - ./configs/zeppelin-env.sh:/opt/zeppelin/conf/zeppelin-env.sh
24    - ./configs/namenode_bashrc.txt:/root/.bashrc
25    - ./configs/namenode_airflow.cfg:/root/airflow/airflow.cfg
26    - ./dags:/root/airflow/dags
27    # - airflow_namenode:/root/airflow
28    # - ./configs/namenode_flume/flume-env.sh:/opt/flume/conf/flume-env.sh
29    # - ./flume_config/flume.conf:/opt/flume/conf/flume.conf
30    # - hadoop_namenode_flume:/opt/flume
31   environment:
32    - CLUSTER_NAME=hadoop-learning
33    - CORE_CONF_fs_defaultFS=hdfs://namenode:9000
34   env_file:
35    - ./hadoop.env
36
```

Figure 1:Le contenu de fichier docker-compose.yml-partie1



```
File Edit Selection View Go Run Terminal Help < -> mise-en-place-d-un-data-lake-master
docker-compose.yml docker-compose spec.json Dockerfile logs environment.yml my_docker_link.json restart client.py
36
37 datanode:
38   image: mrugankray/datanode-python:1.0
39   container_name: datanode
40   restart: always
41   volumes:
42    - hadoop_datanode:/hadoop/dfs/data
43    - hadoop_datanode_conda:/root/anaconda
44    - ./configs/datanode_bashrc.txt:/root/.bashrc
45   environment:
46    SERVICE_PRECONDITION: "namenode:9870"
47    CORE_CONF_fs_defaultFS: "hdfs://namenode:9000"
48   ports:
49    - "9864:9864"
50   env_file:
51    - ./hadoop.env
52
53 resourcemanager:
54   image: mrugankray/resourcemanager-python:1.0
55   container_name: resourcemanager
56   restart: always
57   volumes:
58    - hadoop_resourcemanager_conda:/root/anaconda
59    - ./configs/resourcemanager_bashrc.txt:/root/.bashrc
60   environment:
61    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"
62   ports:
63    - "8088:8088"
64   env_file:
65    - ./hadoop.env
66
67 nodemanager:
68   image: mrugankray/nodemanager-python:1.0
69   container_name: nodemanager
70   restart: always
71   volumes:
72    - hadoop_nodemanager_conda:/root/anaconda
```

Figure 2:Le contenu de fichier docker-compose.yml-partie2

```
File Edit Selection View Go Run Terminal Help < - > mise-en-place-d-un-data-lake-master
docker-compose.yml  apache_beans_compose.yml  test.py  sparkUI.txt  Dockerfile  Hadoop.py  environment.yml  my_kafka_zk.json  shadash.sh  nbt.py  client.py

67  nodemanager:
68    image: mrugankray/nodemanager-python:1.0
69    container_name: nodemanager
70    restart: always
71    volumes:
72      - hadoop_nodemanager_conda:/root/anaconda
73      - ./configs/nodemanager_bashrc.txt:/root/.bashrc
74    environment:
75      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8888"
76    ports:
77      - "8042:8042"
78      - "19888:19888" # to access job history
79    env_file:
80      - ./hadoop.env
81
82  historyserver:
83    image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
84    container_name: historyserver
85    restart: always
86    environment:
87      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8888"
88    ports:
89      - "8188:8188"
90    volumes:
91      - hadoop_historyserver:/hadoop/yarn/timeline
92    env_file:
93      - ./hadoop.env
94
95  # Setting Kafka cluster #
96  zookeeper:
97    image: confluentinc/cp-zookeeper:5.4.0
98    hostname: zookeeper
99    container_name: zookeeper
100   depends_on:
101     - namenode
102     - datanode
103
```

Figure 3:Le contenu de fichier docker-compose.yml-partie3

```
File Edit Selection View Go Run Terminal Help < - > mise-en-place-d-un-data-lake-master
docker-compose.yml  apache_beans_compose.yml  test.py  sparkUI.txt  Dockerfile  Hadoop.py  environment.yml  my_kafka_zk.json  shadash.sh  nbt.py  client.py

231  postgres:
232    container_name: postgres
233    image: postgres:latest
234    ports:
235      - "5432:5432"
236    volumes:
237      - metadata_data:/var/lib/postgresql/data
238    environment:
239      - POSTGRES_PASSWORD=foolishPassword
240      - POSTGRES_USER=druid
241      - POSTGRES_DB=druid
242
243  # Need 3.5 or later for container nodes
244  zookeeper2:
245    container_name: zookeeper2
246    image: zookeeper:3.5.10
247    ports:
248      - "2182:2182"
249    environment:
250      - ZOO_MY_ID=1
251
252  coordinator:
253    image: apache/druid:28.0.0
254    container_name: coordinator
255    volumes:
256      - druid_shared:/opt/shared
257      - coordinator_var:/opt/druid/var
258    depends_on:
259      - zookeeper2
260      - postgres
261      - namenode
262      - datanode
263    ports:
264      - "8072:8072"
265    command:
266      - coordinator
```

Figure 4:Le contenu de fichier docker-compose.yml-partie4

```
File Edit Selection View Go Run Terminal Help < - > mise-en-place-d-un-data-lake-master
docker-compose.yml  test.py  speckle.mn  Dockerfile  buildspec.yml  environment.yml  my_druid_var.json  dist  reset  client.py
270 broker:
271   image: apache/druid:28.0.0
272   container_name: broker
273   volumes:
274     - broker_var:/opt/druid/var
275   depends_on:
276     - zookeeper2
277     - postgres
278     - coordinator
279   ports:
280     - "8076:8076"
281   command:
282     - broker
283   env_file:
284     - environment
285
286 historical:
287   image: apache/druid:28.0.0
288   container_name: historical
289   volumes:
290     - druid_shared:/opt/shared
291     - historical_var:/opt/druid/var
292   depends_on:
293     - zookeeper2
294     - postgres
295     - coordinator
296   ports:
297     - "8019:8019"
298   command:
299     - historical
300   env_file:
301     - environment
302
303 middlemanager:
304   image: apache/druid:28.0.0
305   container_name: middlemanager
306   volumes:
```

Figure 5:Le contenu de fichier docker-compose.yml-partie5

```
File Edit Selection View Go Run Terminal Help < - > mise-en-place-d-un-data-lake-master
docker-compose.yml  test.py  speckle.mn  Dockerfile  buildspec.yml  environment.yml  my_druid_var.json  dist  reset  client.py
303 middlemanager:
304   image: apache/druid:28.0.0
305   container_name: middlemanager
306   volumes:
307     - druid_shared:/opt/shared
308     - middle_var:/opt/druid/var
309   depends_on:
310     - zookeeper2
311     - postgres
312     - coordinator
313   ports:
314     - "8091:8091"
315     - "8100-8105:8100-8105"
316   command:
317     - middleManager
318   env_file:
319     - environment
320
321 router:
322   image: apache/druid:28.0.0
323   container_name: router
324   volumes:
325     - router_var:/opt/druid/var
326   depends_on:
327     - zookeeper2
328     - postgres
329     - coordinator
330     - namenode
331     - datanode
332   ports:
333     - "8888:8888"
334   command:
335     - router
336   env_file:
337     - environment
338
```

Figure 6:Le contenu de fichier docker-compose.yml-partie6

```

341 volumes:
342   hadoop_namenode:
343     hadoop_namenode_conda:
344     hadoop_namenode_spark:
345     hadoop_namenode_zookeeper:
346     hadoop_namenode_flume:
347     hadoop_datanode:
348     hadoop_datanode_conda:
349     hadoop_resourcemanager_conda:
350     hadoop_nodemanager_conda:
351     hadoop_historyserver:
352     airflow_namenode:
353     zookeeper_data:
354     zookeeper_log:
355     kafka_broker:
356     hadoop_hive_server_sqoop:
357     metadata_data: {}
358     middle_var: {}
359     historical_var: {}
360     broker_var: {}
361     coordinator_var: {}
362     router_var: {}
363     druid_shared: {}
364
365
366
367
368
369

```

Figure 7:Les volumes des conteneurs

2. Exécution des conteneurs

Pour lancer les conteneurs spécifiés dans votre fichier docker-compose.yml et les exécuter en mode arrière-plan, vous devez utiliser la commande suivante : **docker-compose up -d**.

```

PS C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master> docker-compose up -d
[+] Building 0.0s (0/0)          docker:default
[+] Running 16/16
  ✓ Container historyserver      Running  0.0s
  ✓ Container zookeeper2        Running  0.0s
  ✓ Container nodemanager       Running  0.0s
  ✓ Container datanode          Running  0.0s
●  ✓ Container namenode         Running  0.0s
  ✓ Container resourcemanager    Running  0.0s
  ✓ Container zookeeper          Running  0.0s
  ✓ Container postgres           Running  0.0s
  ✓ Container coordinator        Running  0.0s
  ✓ Container middlemanager      Running  0.0s
  ✓ Container hive-metastore-postgresql  Running  0.0s
  ✓ Container hive-metastore     Running  0.0s
  ✓ Container broker             Running  0.0s
  ✓ Container historical         Running  0.0s
  ✓ Container router             Running  0.0s
  ✓ Container hive-server        Running  0.0s

```

Figure 8:Exécution des conteneurs

Une fois les conteneurs démarrés, vous avez la possibilité d'afficher les identifiants des conteneurs actifs et leurs ports respectifs en exécutant la commande : **docker ps**.

```

PROBLEMS OUTPUT TERMINAL PORTS POSTMAN CONSOLE COMMENTS DEBUG CONSOLE

PS C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
e8c3934f797c apache/druid:28.0.0 "druid.sh middleMan..." 3 hours ago Up About a minute 0.0.0.0:8091->8091/tcp, 0.0.0.0:8100-810
5->8100-8105/tcp
978944d9321a apache/druid:28.0.0 "druid.sh broker" 3 hours ago Up About a minute 0.0.0.0:8076->8076/tcp
99fd7f6c2254 apache/druid:28.0.0 "/druid.sh historical" 3 hours ago Up About a minute 0.0.0.0:8019->8019/tcp
e5ca6fa928d3 apache/druid:28.0.0 "/druid.sh router" 3 hours ago Up About a minute 0.0.0.0:8888->8888/tcp
b0b8d78572b8 apache/druid:28.0.0 "/druid.sh coordinator" 3 hours ago Up About a minute 0.0.0.0:8072->8072/tcp
047c46c7165e mrugankray/hive-server-sqoop:1.0 "/druid.sh entrypoint.sh /bin/.." 3 days ago Up About a minute 0.0.0.0:10000->10000/tcp, 10002/tcp
151f6e9ed6dd confluentinc/cp-zookeeper:5.4.0 "/etc/confluent/docker..." 3 days ago Up About a minute 2888/tcp, 0.0.0.0:2181->2181/tcp, 3888-tcp
cp 2f8cc30f216c bde2020/hive:2.3.2-postgresql1-metastore "/etc/confluent/docker..." 3 days ago Up About a minute 10000/tcp, 0.0.0.0:9083->9083/tcp, 10002-
/tcp
5099ab5cf396 mrugankray/resourcemanager-python:1.0 "/entrypoint.sh /run..." 3 days ago Up 17 seconds (health: starting) 8042/tcp, 0.0.0.0:8088->8088/tcp
resourcemanager
1b3fec49d16a mrugankray/namenode-spark-airflow-flume-zeppelin:1.1 "/entrypoint.sh /stat..." 3 days ago Up About a minute (healthy) 0.0.0.0:4040->4040/tcp, 0.0.0.0:8080-808
2->8080-8082/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp
namenode
cfedbce86d99 mrugankray/nodemanager-python:1.0 "/entrypoint.sh /run..." 3 days ago Up About a minute (healthy) 0.0.0.0:8042->8042/tcp, 0.0.0.0:19888->1
9888/tcp, 8088/tcp
1abf980d931a mrugankray/datanode-python:1.0 "/entrypoint.sh /run..." 3 days ago Up About a minute (healthy) 0.0.0.0:9864->9864/tcp
datanode
9d93c540c01c bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8 "/entrypoint.sh /run..." 3 days ago Up About a minute (healthy) 0.0.0.0:8188->8188/tcp
historyserver
0e6593e9fb4 zookeeper:3.5.10 "/docker-entrypoint..." 3 days ago Up About a minute 2181/tcp, 2888/tcp, 3888/tcp, 8080/tcp,
0.0.0.0:2182->2182/tcp
zookeeper2
1add00e8406 bde2020/hive-metastore-postgresql:2.3.0 "/docker-entrypoint..." 3 days ago Up About a minute 5432/tcp
hive-metastore-postgresql
bf08ea0754cc postgres:latest "/docker-entrypoint..." 3 days ago Up About a minute 0.0.0.0:5432->5432/tcp
postgres

○ PS C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master>

```

Ou bien vous pouvez consulter l'interface de Docker Desktop.

The screenshot shows the Docker Desktop application window. On the left, there's a sidebar with navigation links: Containers, Images, Volumes, Dev Environments (BETA), Docker Scout, and Learning center. Below these are sections for Extensions and Add Extensions. The main area is titled 'Containers' and contains a search bar and a chart showing Container CPU usage (28.50% / 400%) and Container memory usage (6.02GB / 7.52GB). A table lists 17 running containers, each with a status icon, name, image, status, CPU usage, port mappings, and last started time. The containers listed are: mise-en-place-d-un-data-lake-r, middlemanager, broker, historical, router, coordinator, hive-server, zookeeper, and hive-metastore.

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
	mise-en-place-d-un-data-lake-r	apache/druid:28.0.0	Running (16/1)	28.5%	8091:8091 ↗ Show all ports (7)	3 minutes ago	⋮ ⚡ ⚡
	middlemanager	e8c3934f797c	Running	0.51%	8091:8091 ↗	5 minutes ago	⋮ ⚡ ⚡
	broker	978944d9321a	Running	5.43%	8076:8076 ↗	5 minutes ago	⋮ ⚡ ⚡
	historical	99fd7f6c2254	Running	0.47%	8019:8019 ↗	5 minutes ago	⋮ ⚡ ⚡
	router	e5ca6fa928d3	Running	0.48%	8888:8888 ↗	5 minutes ago	⋮ ⚡ ⚡
	coordinator	b0b8d78572b8	Running	0.73%	8072:8072 ↗	5 minutes ago	⋮ ⚡ ⚡
	hive-server	047c46c7165e	Running	0.7%	10000:10000 ↗	5 minutes ago	⋮ ⚡ ⚡
	zookeeper	151f6e9ed6dd	Running	0.1%	2181:2181 ↗	5 minutes ago	⋮ ⚡ ⚡
	hive-metastore						

Figure 9:interface de docker Desktop

4. Configuration des api New York Times et Yahoo Finance

1. Api New York Times

Vous devez consulter le lien suivant : <https://developer.nytimes.com/apis>

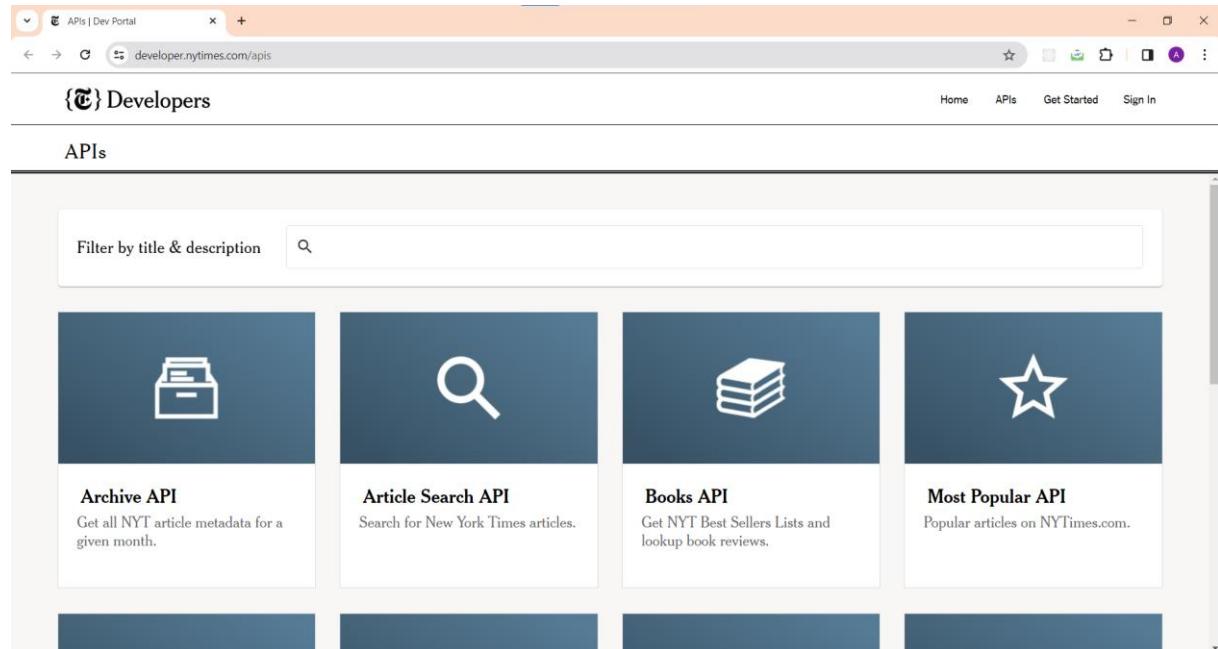
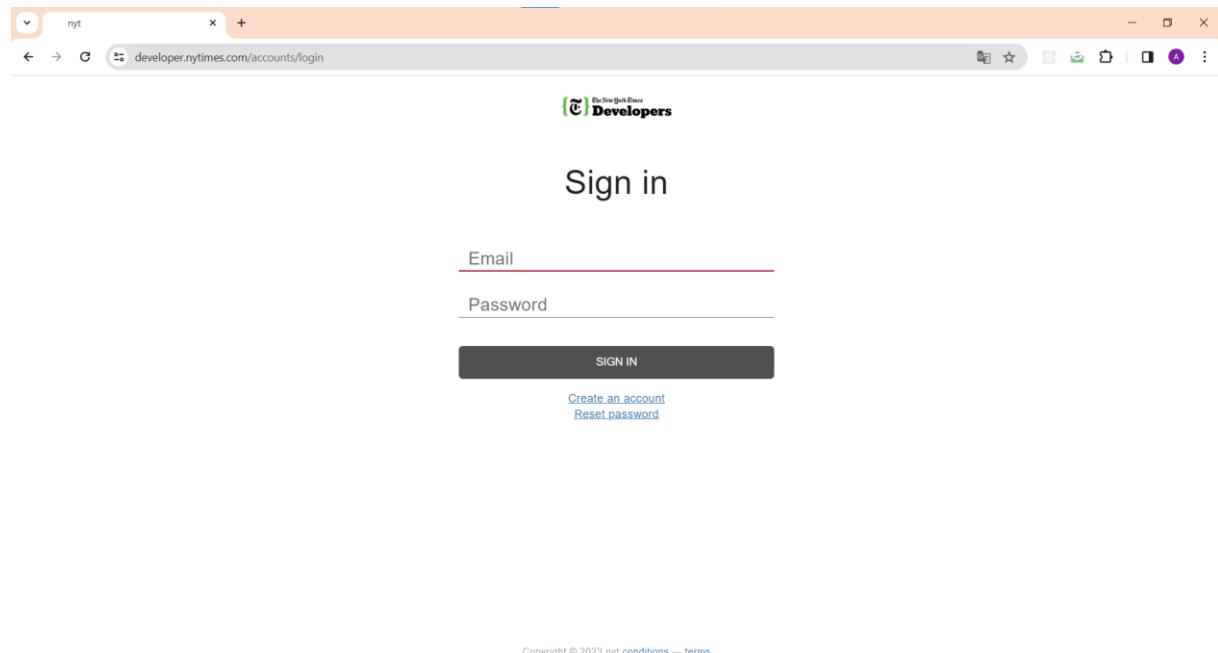


Figure 10: Interface des apis New York Times

Puis vous cliquez sur Sign In et après vous serez amené à créer un nouveau compte



The screenshot shows a web browser window with the URL developer.nytimes.com/accounts/create. The page title is "Create your account". It contains four input fields: "First Name", "Last Name", "Email", and "Password". Below the password field is a checkbox for agreeing to the "conditions and the terms". A "Create Account" button is at the bottom, and a "Sign in" link is below it.

Une fois vous terminez, vous serez dirigé vers la page suivante :

The screenshot shows the New York Times Developers homepage. The main heading is "The New York Times Developer Network" with the tagline "All the APIs Fit to Post". There is a "GET STARTED" button. Below the main banner are three sections: "Get Started", "APIs", and "FAQ". The "Get Started" section says "Learn how to sign up for an API key.", the "APIs" section says "Learn about and try out NYT's APIs.", and the "FAQ" section says "Get answers to frequently asked questions." The top navigation bar includes links for "Home", "APIs", "Get Started", and a user account link.

Vous choisissez Apis, ensuite cliquez sur Archive API

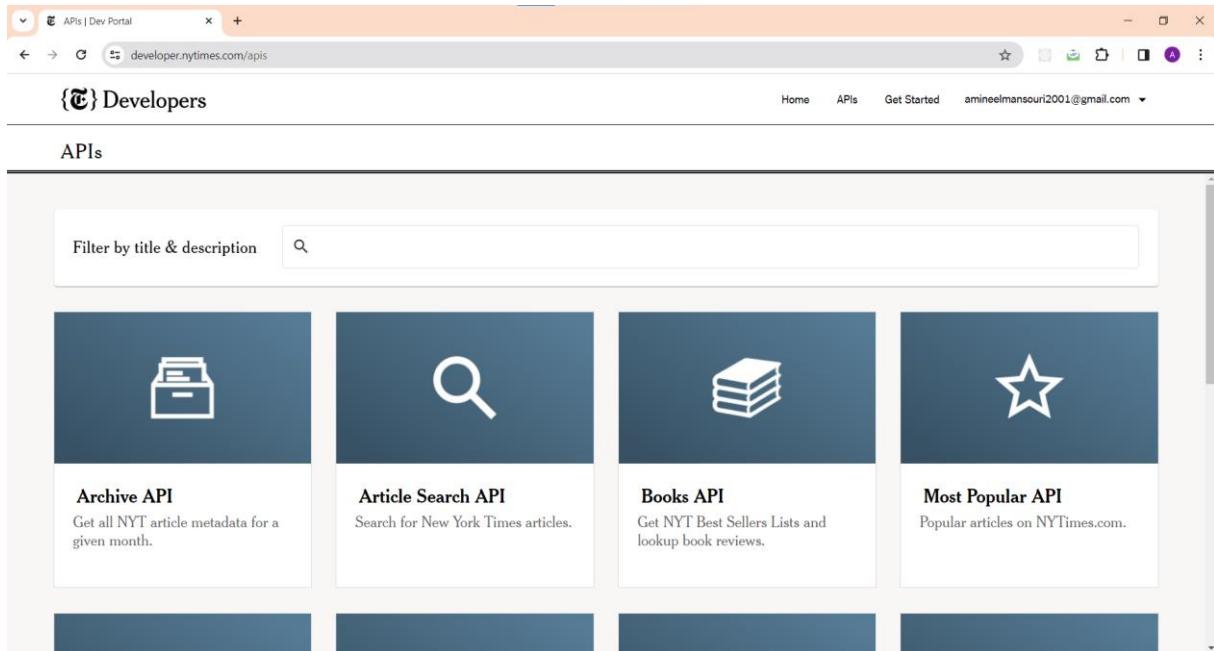


Figure 11:Apis New York Times

The screenshot shows the "Archive" API documentation page. On the left, there's a sidebar with "ARCHIVE" selected, followed by "Overview", "PATHS", and "COMPONENTS" sections. The "COMPONENTS" section lists "Article", "Byline", "Headline", "Keyword", "Multimedia", and "Person". The main content area is titled "Archive" and contains the following information:

- Description:** The Archive API returns an array of NYT articles for a given month, going back to 1851. Its response fields are the same as the Article Search API. The Archive API is very useful if you want to build your own database of NYT article metadata. You simply pass the API the year and month and it returns a JSON object with all articles for that month. The response size can be large (~20mb).
- Example Call:** `/year/(month).json`
- Resource Types:** A list of components: Article, Byline, Headline, Keyword, Multimedia, Person.

Figure 12:Archive Api pour les articles de New York Times

Vous cliquez sur votre adresse mail et vous choisissez **apps**

The screenshot shows the homepage of the New York Times Developer Network. At the top right, there is a user profile for "amineelmansouri2001@gmail.com" with options for "Apps" and "Sign Out". The main header reads "{T} Developers" and "The New York Times Developer Network". Below the header, a banner says "All the APIs Fit to Post" with a "GET STARTED" button. The page features three main sections: "Get Started" (with a link to https://developer.nytimes.com/my-apps), "APIs" (with a link to https://developer.nytimes.com/apis), and "FAQ" (with a link to https://developer.nytimes.com/faq).

Cliquez sur New App

The screenshot shows the "My Apps" section of the developer portal. At the top right, there is a user profile for "amineelmansouri2001@gmail.com". The main header reads "{T} Developers" and "My Apps". Below the header, there is a table titled "My Apps" showing three existing applications:

Name	Description	Created
NYT_archive_api		Dec 18, 2023
NYT_article_real_time		Dec 18, 2023
myapp	desc	Dec 10, 2023

Par la suite vous ecrivez le nom de votre app ainsi qu'une description de cette app et vous cliquez sur l'action enable pour activer l'api **Archive API** Puis cliquez sur save pour enregistrer l'app.

The screenshot shows the 'New App' creation interface on the NYTimes Dev Portal. The 'Overview' section contains fields for 'App Name' (with 'NYT_archive_api' typed in) and 'Description'. Below this is a table titled 'APIs *' listing four APIs: 'Archive API', 'Article Search API', 'Books API', and 'Most Popular API', each with an 'Enable' button. At the bottom are 'CANCEL', 'CLEAR', and 'SAVE' buttons.

Figure 13: La création d'une nouvelle app

En fin, voilà l'api key a été générer avec succès vous pouvez l'utiliser par la suite pour intercepter les données de New Yourk Times .

The screenshot shows the configuration interface for the 'NYT_archive_api' on the NYTimes Dev Portal. It displays the app name ('NYT_archive_api'), description, and app ID ('e51dca52-1ddb-4940-8ce0-2d7b0f15a997'). Below is a table for 'API Keys' showing one key entry: 'WEJG5L43Blt4E86fvVgWAAsvgeeeUV54' (Secret: 'Show secret'), Status: 'Active', Created: 'Dec 18, 2023, 11:10 AM', Expires: 'never', and Actions: 'Revoke'. At the bottom are 'DELETE', 'RESET', and 'SAVE' buttons.

2. Yahoo finance Api

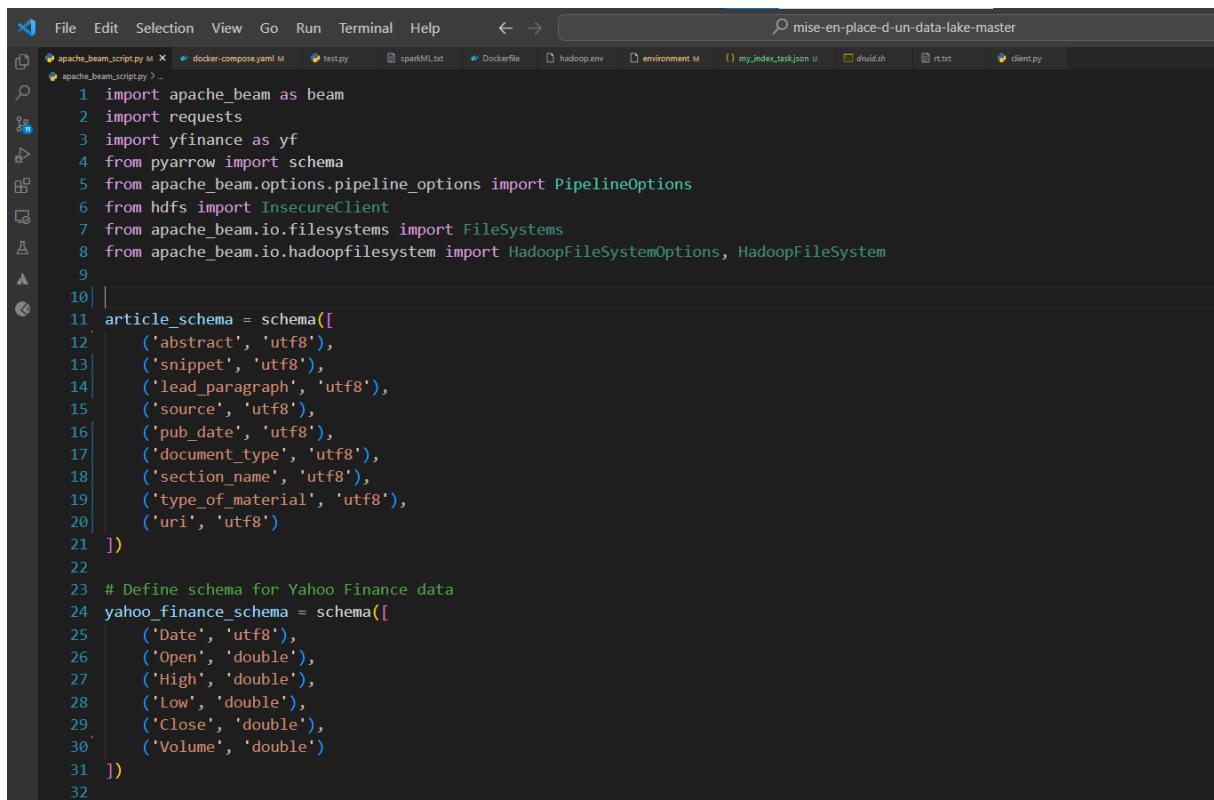
Yahoo Finance propose une API qui permet d'accéder à diverses informations financières, notamment les cotations boursières, les informations sur les indices, les données historiques, etc. Pour interagir avec l'API Yahoo Finance en utilisant Python, vous pouvez utiliser des bibliothèques telles que **yfinance**.

Il suffit d'Installer la bibliothèque **yfinance**, via la commande suivante :

⇒ pip install yfinance

5. Interception des données des 2 Apis en utilisant Apache Beam

Ce code python utilise Apache Beam pour créer un pipeline de traitement de données. Il définit deux schémas de données, l'un pour les articles d'actualité de New York Times et l'autre pour les informations financières de Yahoo Finance, en utilisant la bibliothèque PyArrow. Le pipeline envisage de récupérer des données d'articles via des requêtes HTTP, ainsi que des données financières de Yahoo Finance, et les traiter simultanément. Les schémas spécifient la structure des données pour garantir une cohérence lors du traitement parallèle dans le pipeline Apache Beam.

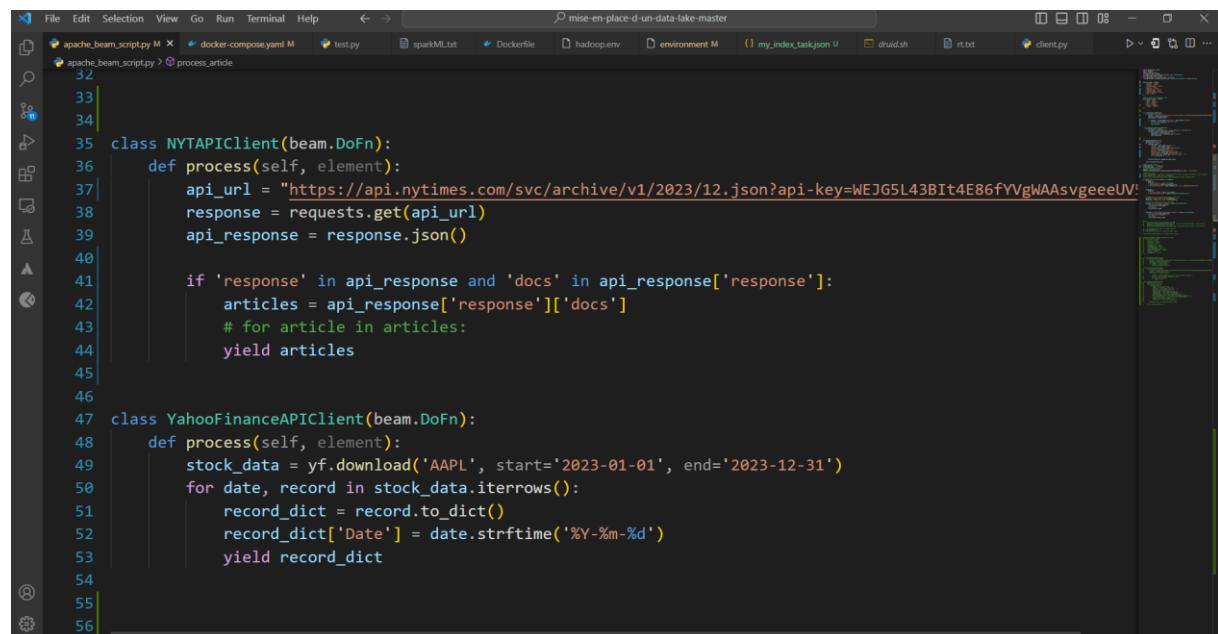


```
File Edit Selection View Go Run Terminal Help mise-en-place-d-un-data-lake-master
apache_beam_script.py M docker-compose.yml M test.py sparkML.txt Dockerfile hadoop-env environment.json my_index_task.json U droid.sh rt.txt client.py
apache_beam_script.py ...
1 import apache_beam as beam
2 import requests
3 import yfinance as yf
4 from pyarrow import schema
5 from apache_beam.options.pipeline_options import PipelineOptions
6 from hdfs import InsecureClient
7 from apache_beam.io.filesystems import FileSystems
8 from apache_beam.io.hadoopfilesystem import HadoopFileSystemOptions, HadoopFileSystem
9
10 |
11 article_schema = schema([
12     ('abstract', 'utf8'),
13     ('snippet', 'utf8'),
14     ('lead_paragraph', 'utf8'),
15     ('source', 'utf8'),
16     ('pub_date', 'utf8'),
17     ('document_type', 'utf8'),
18     ('section_name', 'utf8'),
19     ('type_of_material', 'utf8'),
20     ('uri', 'utf8')
21 ])
22
23 # Define schema for Yahoo Finance data
24 yahoo_finance_schema = schema([
25     ('Date', 'utf8'),
26     ('Open', 'double'),
27     ('High', 'double'),
28     ('Low', 'double'),
29     ('Close', 'double'),
30     ('Volume', 'double')
31 ])
32
```

Figure 14:Interception des données-partiel

Ce code utilise Apache Beam pour créer un pipeline de traitement de données.

- Deux classes (**NYTAPIClient** et **YahooFinanceAPIClient**) interrogent respectivement l'API du New York Times et Yahoo Finance pour récupérer des données.
- La fonction **process_article** extrait des champs spécifiques des articles du New York Times.
- Le pipeline traite les données en parallèle, imprime des informations de débogage, puis écrit les résultats dans des fichiers Parquet.
- Les chemins de sortie (**output_path_nyt** et **output_path_yahoo**) et les options du pipeline sont configurés.



The screenshot shows a code editor with multiple tabs open. The active tab contains Python code for an Apache Beam pipeline. The code defines two classes: **NYTAPIClient** and **YahooFinanceAPIClient**, both inheriting from **beam.DoFn**. The **NYTAPIClient** class has a **process** method that sends a GET request to the New York Times API and yields the extracted articles. The **YahooFinanceAPIClient** class uses the **yf** library to download stock data for 'AAPL' from January 1, 2023, to December 31, 2023, and yields the data as dictionaries. The code editor interface includes a top menu bar, a toolbar, and a sidebar with various icons.

```
File Edit Selection View Go Run Terminal Help ← → mise-en-place-d-un-data-lake-master
apache_beam_script.py M docker-compose.yaml M test.py sparkML.txt Dockerfile hadoop.env environment M my_index_task.json U druid.sh rt.txt client.py D ...
32
33
34
35 class NYTAPIClient(beam.DoFn):
36     def process(self, element):
37         api_url = "https://api.nytimes.com/svc/archive/v1/2023/12.json?api-key=WEJG5L43BIt4E86fYVgWAAsvgeeeUV"
38         response = requests.get(api_url)
39         api_response = response.json()
40
41         if 'response' in api_response and 'docs' in api_response['response']:
42             articles = api_response['response']['docs']
43             # for article in articles:
44             yield articles
45
46
47 class YahooFinanceAPIClient(beam.DoFn):
48     def process(self, element):
49         stock_data = yf.download('AAPL', start='2023-01-01', end='2023-12-31')
50         for date, record in stock_data.iterrows():
51             record_dict = record.to_dict()
52             record_dict['Date'] = date.strftime('%Y-%m-%d')
53             yield record_dict
54
55
56
```

Figure 15: Interception des données-partie2

```
File Edit Selection View Go Run Terminal Help < > mise-en-place-d-un-data-lake-master
apache_beam_script.py M docker-compose.yaml M test.py sparkML.txt Dockerfile hadoop-env environment M my_index_task.json U druid.sh rt.txt client.py < > ...
```

```
55
56
57 def process_article(articles):
58     selected_fields_list = []
59     for article in articles:
60         selected_fields = {
61             'abstract': article.get('abstract', ''),
62             'snippet': article.get('snippet', ''),
63             'lead_paragraph': article.get('lead_paragraph', ''),
64             'source': article.get('source', ''),
65             'pub_date': article.get('pub_date', ''),
66             'document_type': article.get('document_type', ''),
67             'section_name': article.get('section_name', ''),
68             'type_of_material': article.get('type_of_material', ''),
69             'uri': article.get('uri', '')
70         }
71         selected_fields_list.append(selected_fields)
72
73     return selected_fields_list
74
75
76 # Output paths in HDFS
77 output_path_nyt = "./test6/nyt"
78 output_path_yahoo = "./test6/yh"
```

Figure 16: Interception des données-partie3

```
File Edit Selection View Go Run Terminal Help < > mise-en-place-d-un-data-lake-master
apache_beam_script.py M docker-compose.yaml M test.py sparkML.txt Dockerfile hadoop-env environment M my_index_task.json U druid.sh rt.txt client.py < > ...
```

```
79 # Apache Beam Pipeline Options
80 options = PipelineOptions()
81 options.view_as(beam.options.pipeline_options.StandardOptions).runner = 'DirectRunner'
82
83 # HDFS Client Setup
84 # hdfs_namenode_host = 'localhost' # Replace with the actual hostname or IP address of your NameNode
85 # hdfs_namenode_port = 9870 # Port du NameNode
86 # hdfs_client = InsecureClient(f'http://{{hdfs_namenode_host}}:{hdfs_namenode_port}')
87
88 with beam.Pipeline(options=options) as pipeline:
89     nyt_data = (
90         pipeline
91         | "Create URL NYT" >> beam.Create([None])
92         | "Get NYT Data" >> beam.ParDo(NYTAPIClient())
93         | "Process NYT Articles" >> beam.FlatMap(lambda result: process_article(result))
94     )
95
96     yahoo_data = (
97         pipeline
98         | "Create URL Yahoo" >> beam.Create([None])
99         | "Get Yahoo Finance Data" >> beam.ParDo(YahooFinanceAPIClient())
100    )
101
102    # Debugging: Print the data before writing to Parquet
103    nyt_data | "Debug Print NYT" >> beam.Map(print)
104    yahoo_data | "Debug Print Yahoo" >> beam.Map(print)
105
106    # Write data to Parquet with explicit schema
107    nyt_data | "Write NYT Data to Parquet" >> beam.io.WriteToParquet(
108        file_path_prefix=output_path_nyt,
109        file_name_suffix=".parquet",
110        num_shards=1,
111        schema=article_schema
112    )
113
114    yahoo_data | "Write Yahoo Finance Data to Parquet" >> beam.io.WriteToParquet(
115        file_path_prefix=output_path_yahoo,
116        file_name_suffix=".parquet",
117        num_shards=1,
118        schema=yahoo_finance_schema
119    )
```

Figure 17: Interception des données-partie4

6. Enregistrement des données brutes sous format Parquet

Les résultats de l'exécution du pipeline Beam sont écrits dans des fichiers Parquet avec des schémas explicites pour les données du New York Times et de Yahoo Finance.

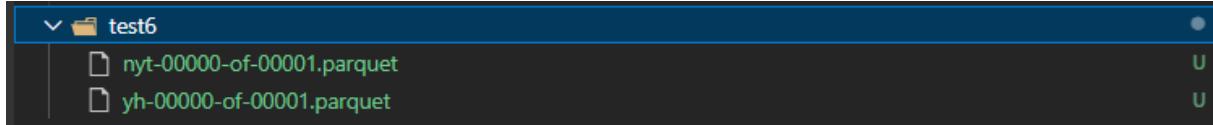


Figure 18: Les fichiers parquet générés

7. Entraînement de plusieurs modèles ML pour sentiment analysis

Voici une description des étapes que vous pourriez suivre pour entraîner plusieurs modèles de sentiment analysis en utilisant Spark ML sur un notebook Zeppelin et enregistrer le meilleur modèle :

1. **Importation des bibliothèques Spark ML :**
 - Débutez par importer les bibliothèques Spark ML nécessaires.
2. **Définition des modèles et de leurs hyperparamètres :**
 - Définissez une liste de modèles de classification tels que Logistic Regression, Naive Bayes et RandomForest, avec leurs hyperparamètres respectifs.
3. **Division du dataset en ensembles d'entraînement et de test :**
 - Divisez le dataset en ensembles d'entraînement et de test en utilisant la méthode **randomSplit**.

```
%pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()
Took 1 sec. Last updated by anonymous at December 17 2023, 12:27:02 PM.

%pyspark
from pyspark.ml.classification import LogisticRegression, NaiveBayes, RandomForestClassifier
from pyspark.ml.Pipeline import Pipeline
from pyspark.ml.feature import Tokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
# Initialiser la session Spark
spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()
Took 1 sec. Last updated by anonymous at December 17 2023, 12:27:08 PM.

+ Add Paragraph

%pyspark
# Charger le dataset dans un DataFrame Spark
# dataset_path = "userInputs/data.csv"
dataset_path = "hdfs://namenode:9000/user/inputs/data.csv"
df = spark.read.option("header", "true").option("inferSchema", "true").csv(dataset path)
Took 15 sec. Last updated by anonymous at December 17 2023, 12:36:55 PM.

%pyspark
# Prétraitement des données
tokenizer = Tokenizer(inputCol="Sentence", outputCol="words")
remover = StopWordsRemover(inputCol="words", outputCol="filtered")
vectorizer = CountVectorizer(inputCol="filtered", outputCol="features")
Took 1 sec. Last updated by anonymous at December 17 2023, 12:37:31 PM.
```

Figure 19:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie1

4. Prétraitement des données avec un pipeline :

- Pour chaque modèle, créez un pipeline comprenant plusieurs étapes : Tokenization des phrases, HashingTF pour la conversion des mots en vecteurs, IDF pour pondérer les termes, StringIndexer pour indexer la variable cible "Sentiment", et l'entraînement du modèle spécifique (Logistic Regression, Naive Bayes, RandomForest).

```
%pyspark
# Liste des modèles avec hyperparamètres
models = [
    (LogisticRegression, {
        "labelCol": "Sentiment",
        "featuresCol": "features",
        "maxIter": 10,
        "regParam": 0.01
    }),
    (NaiveBayes, {
        "labelCol": "Sentiment",
        "featuresCol": "features",
        "smoothing": 1.0
    }),
    (RandomForestClassifier, {
        "labelCol": "Sentiment",
        "featuresCol": "features",
        "numTrees": 10,#50
        "maxDepth": 5
    })
]
# Diviser le dataset en ensembles d'entraînement et de test
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)
train_data = train_data.repartition(8)
test_data = test_data.repartition(8)
Took 0 sec. Last updated by anonymous at December 17 2023, 12:37:40 PM. (outdated)

%pyspark
print(df.columns)
['Sentence', 'Sentiment']
Took 0 sec. Last updated by anonymous at December 17 2023, 12:37:47 PM.
```

Figure 20:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie2

```

spark_ml_model - Zeppelin
localhost:8082/#/notebook/2JK9QJUNB

Zeppelin Notebook Job
spark_ml_model | Head | Search | anonymous | default

['Sentence', 'Sentiment']

Took 0 sec. Last updated by anonymous at December 17 2023, 12:37:47 PM.

%spark
df.show(10)
+-----+
| Sentence|Sentiment|
+-----+
|The GeoSolutions ...| positive|
|$ESI on lows, dow...| negative|
|For the last quart...| positive|
|According to the ...| neutral|
|The Swedish buyout...| neutral|
|$SPY wouldn't be ...| positive|
|Shell's $70 Billi...| negative|
|SSH COMMUNICATION...| negative|
|Kone 's net sales...| positive|
|The Stockmann dep...| neutral|
+-----+
only showing top 10 rows

Took 0 sec. Last updated by anonymous at December 17 2023, 12:37:51 PM.

SPARK JOB FINISHED

```

Figure 21:code d'entraînement de plusieurs Modèles ML pour sentiment analysis-partie3

5. Entraînement et évaluation des modèles :

- Utilisez un échantillon de 10% des données d'entraînement pour accélérer le processus. Entraînez et évaluez les modèles sur l'ensemble de test, puis imprimez l'exactitude de chaque modèle.

```

spark_ml_model - Zeppelin
localhost:8082/#/notebook/2JK9QJUNB

Took 0 sec. Last updated by anonymous at December 17 2023, 12:37:51 PM.

%spark
from pyspark.ml.feature import Tokenizer, HashingTF, IDF, StringIndexer
from pyspark.ml.classification import NaiveBayes, RandomForestClassifier
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

Took 0 sec. Last updated by anonymous at December 17 2023, 12:38:07 PM.

%spark
# Entrainer et évaluer les modules
best_model = None
best_accuracy = 0.0

for model_class, param_grid in models:
    print(f"Créer le pipeline")
    tokenizer = Tokenizer(inputCol="Sentence", outputCol="words")
    hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=50)
    idf = IDF(inputCol="rawFeatures", outputCol="features")
    classifier = model_class()

    indexer = StringIndexer(inputCol="Sentiment", outputCol="label_indexed") # Use "Sentiment" instead of "Sentence"
    classifier = classifier.setInputCol("label_indexed").setOutputCol("features")

    pipeline = Pipeline(stages=[tokenizer, hashingTF, idf, indexer, classifier])
    pipeline.fit(param_grid)

    print(f"Training {model_class.__name__} model...")
    model = pipeline.fit(train_data.sample(fraction=0.1, seed=42))
    print(f"({model_class.__name__}) model trained successfully.")

    predictions = model.transform(test_data.sample(fraction=0.1, seed=42))

    evaluator = MulticlassClassificationEvaluator(labelCol="label_indexed", predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print(f"({model_class.__name__}) Accuracy: {accuracy}")

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model

print(f"Best model with accuracy: {best_accuracy}")

Training LogisticRegression model...
LogisticRegression model trained successfully.
LogisticRegression Accuracy: 0.6295981919711312
Training NaiveBayes model...
NaiveBayes model trained successfully.
NaiveBayes Accuracy: 0.40935655573771
Training RandomForestClassifier model...
RandomForestClassifier model trained successfully.
RandomForestClassifier Accuracy: 0.69344262295982
Best model with accuracy: 0.69344262295982

Took 35 ms. Last updated by anonymous at December 17 2023, 12:38:08 PM.

SPARK JOB FINISHED

```

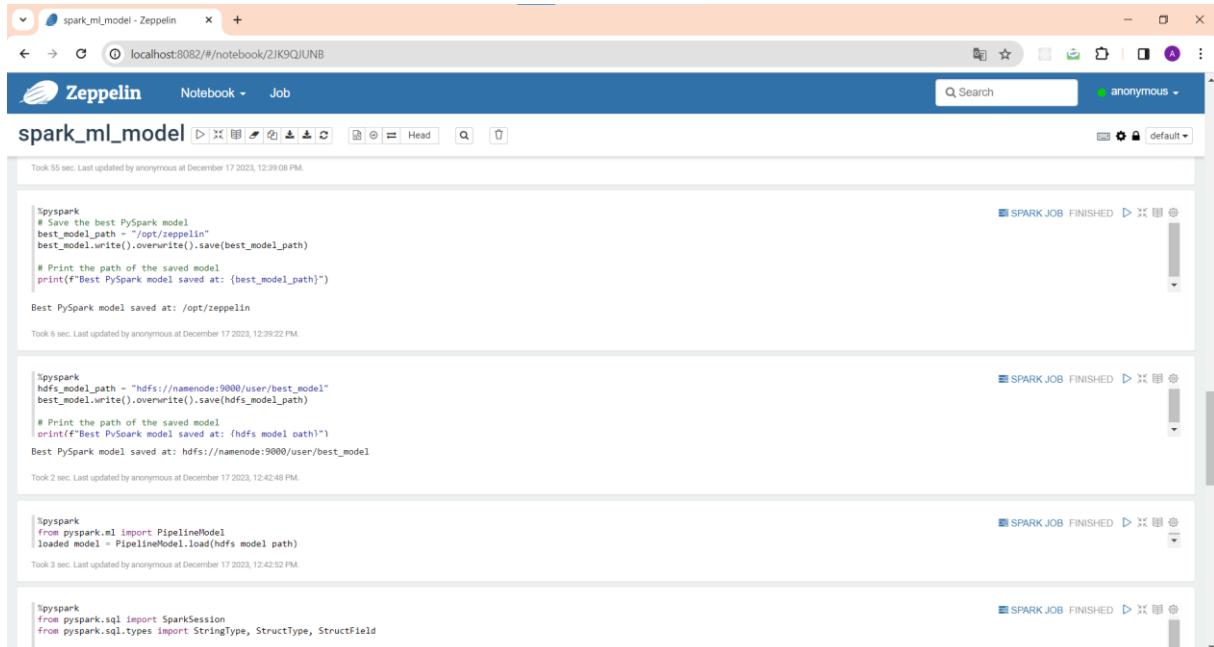
Figure 22:Entraînement et évaluation des modèles

6. Sélection du meilleur modèle :

- Suivez une approche de sélection du meilleur modèle en comparant les exactitudes et enregistrez le modèle avec la meilleure précision.

7. Enregistrement du meilleur modèle sur HDFS :

- Enregistrez le meilleur modèle sur le système de fichiers distribué HDFS en utilisant le chemin spécifié.



```
#pyspark
# Save the best PySpark model
best_model_path = "/opt/zeppelin"
best_model.write().overwrite().save(best_model_path)

# Print the path of the saved model
print(f"Best PySpark model saved at: {best_model_path}")

Best PySpark model saved at: /opt/zeppelin
Took 6 sec. Last updated by anonymous at December 17 2023, 12:39:08 PM.

#pyspark
Hdfs_model_path = "hdfs://namenode:9000/user/best_model"
best_model.write().overwrite().save(hdfs_model_path)

# Print the path of the saved model
print(f"Best PySpark model saved at: {hdfs_model_path}")
Best PySpark model saved at: hdfs://namenode:9000/user/best_model
Took 2 sec. Last updated by anonymous at December 17 2023, 12:42:48 PM.

#pyspark
from pyspark.ml import PipelineModel
loaded_model = PipelineModel.load(hdfs_model_path)
Took 3 sec. Last updated by anonymous at December 17 2023, 12:42:52 PM.
```

Figure 23:Enregistrement du meilleur modèle sur HDFS

8. Chargement du modèle enregistré :

- Chargez le modèle enregistré à partir de HDFS pour une utilisation ultérieure.

9. Utilisation du modèle pour faire des prédictions sur de nouvelles données :

- Créez un Spark DataFrame avec de nouvelles données de texte et utilisez le modèle chargé pour faire des prédictions sur ces nouvelles données.

```

spark
from pyspark.ml import PipelineModel
loaded_model = PipelineModel.load("nmls_model_path")

```

Took 3 sec. Last updated by anonymous at December 17 2023, 12:42:52 PM.

```

# Import required libraries
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType, StructType, StructField

# Create a Spark session
spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

# Sample new data
data = [
    ("This is a positive sentence.",),
    ("Negative sentiment is not good.,"),
    ("I feel not happy today.,"),
    ("Sad news makes me unhappy.,")
]

# Define the schema for the new data
schema = StructType([StructField("Sentence", StringType(), True)])

# Create a DataFrame for new data
new_data = spark.createDataFrame(data, schema=schema)

# Show the new data
new_data.show(truncate=False)

```

-----+
|Sentence|
+-----+
|This is a positive sentence.|
|Negative sentiment is not good.|
|I feel not happy today.|
|Sad news makes me unhappy.|
+-----+

Took 1 sec. Last updated by anonymous at December 17 2023, 12:42:52 PM.

Figure 24: Utilisation du modèle pour faire des prédictions sur de nouvelles données

10. Affichage des résultats des prédictions :

- Définissez une fonction pour mapper les prédictions numériques à des libellés de sentiment (positif, négatif, neutre). Appliquez cette fonction aux prédictions et sélectionnez les colonnes pertinentes pour l'affichage des résultats.

```

spark
from pyspark.sql.functions import udf
# Make a prediction on the new data
new_predictions = loaded_model.transform(new_data)

# Define a function to map numeric predictions to sentiment labels
def map_sentiment_label(prediction):
    if prediction == 0.0:
        return 'positive'
    elif prediction == 1.0:
        return 'negative'
    else:
        return 'neutral'

# Create a user-defined function (UDF) for the mapping function
map_sentiment_udf = udf(map_sentiment_label, StringType())

# Apply the mapping function to the 'prediction' column and create a new column 'sentiment'
new_predictions = new_predictions.withColumn('sentiment', map_sentiment_udf(new_predictions['prediction']))

# Select the relevant columns for display
result = new_predictions.select('Sentence', 'sentiment', 'probability')

# Show the result
result.show(truncate=False)

```

-----+|Sentence|sentiment|probability|
+-----+|This is a positive sentence.|positive|[0.5451254335541086, 0.30616433123557263, 0.1487102352103188]|
Negative sentiment is not good.	negative	[0.17318334440001086, 0.5357608872499746, 0.2904050581499248]
I feel not happy today.	negative	[0.1665846941764669, 0.543959838671346, 0.290355467138639876]
Sad news makes me unhappy.	negative	[0.13223932301167857, 0.5323029718444341, 0.33545770514388734]
+-----+

Took 2 sec. Last updated by anonymous at December 17 2023, 12:42:56 PM.

Figure 25: Affichage des résultats des prédictions

8. La lecture des données enregistrées avec Spark SQL

Les étapes que vous devez suivre en utilisant PySpark pour lire et traiter des données à partir de fichiers Parquet :

- **Initialisez la session Spark**
- **Lisez les fichiers Parquet :**
 - ✓ Définissez les chemins des fichiers Parquet du New York Times (nyt_path) et de Yahoo Finance (yahoo_path).
 - ✓ Utilisez la méthode spark.read.parquet() pour lire ces fichiers et créer les DataFrames correspondants (nyt_df et yahoo_df).

```

spark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

# Define the path to Parquet files in HDFS
nyt_path = "hdfs://namenode:9000/user/inputs/nyt-00000-of-00001.parquet"
yahoo_path = "hdfs://namenode:9000/user/inputs/yh-00000-of-00001.parquet"

# Read Parquet files into DataFrames
nyt_df = spark.read.parquet(nyt_path)
# Rename columns with invalid characters using aliases
nyt_df = nyt_df.select(
    col("abstract"),
    col("body"),
    col("lead_paragraph"),
    col("source"),
    col("text"),
    col("document_type"),
    col("section_name"),
    col("type_of_material"),
    col("url"),
)

# Register DataFrames as temporary tables
nyt_df.createOrReplaceTempView("nyt_data")

# Read Yahoo Finance Parquet file
yahoo_df = spark.read.parquet(yahoo_path)
# Rename columns with invalid characters using aliases
yahoo_df = yahoo_df.select(
    col("Date"),
    col("Open"),
    col("High"),
    col("Low"),
    col("Close"),
    col("Volume")
)

```

Figure 26:La lecture des données enregistrées avec Spark SQL

- **Enregistrez les DataFrames en tant que tables temporaires :**
 - ✓ Utilisez la méthode **createOrReplaceTempView** pour enregistrer les DataFrames en tant que tables temporaires dans Spark, permettant l'exécution de requêtes SQL.
- **Exécutez des requêtes SQL Spark :**
 - ✓ Utilisez la fonction **spark.sql()** pour exécuter des requêtes SQL sur les tables temporaires. Par exemple, sélectionnez toutes les colonnes des tables **nyt_data** et **yahoo_data**.
- **Affichez les résultats :**
 - ✓ Utilisez la méthode **show()** pour afficher les résultats des requêtes SQL.

```

# Register DataFrame as a temporary table
yahoo_df.createOrReplaceTempView("yahoo_data")

# Perform Spark SQL queries to analyze or transform the data
result_myt = spark.sql("""
    SELECT *
    FROM myt
""")

result_yahoo = spark.sql("""
    SELECT *
    FROM yahoo_data
""")

# Display the results
result_myt.show()
result_yahoo.show()

# spark.stop()

```

Figure 27:L'affichage des résultats des requêtes SQL-partie1

Date	Open	High	Low	Close	Volume
2023-01-01 138.2799977929688 139.8999339948481 142.169981699531 125.869996048242 11.12117981					
2023-01-04 126.889993896444 128.660083621093 125.0800081316540 126.3600061851561 8.9113647					
2023-01-05 127.12999725341797 127.769996468064 124.76000213623047 125.019996436464 8.0962787					
2023-01-06 126.01800213623047 126.289993261132 126.8899931994664 126.1099931172375 8.773477					
2023-01-07 126.01800213623047 126.289993261132 126.8899931994664 126.1099931172375 8.773477					
2023-01-10 138.2599945063594 132.2599945063594 128.1139991171875 130.2299957253901 6.309827					
2023-01-11 131.25 133.5099945063594 130.460000713867 133.4000054330446 6.9450987					
2023-01-12 131.8000048828125 134.2599945063594 131.44000024146825 133.41000362320981 7.137967					
2023-01-13 132.0299977929688 134.019998169453 131.6600004621093 133.759994683594 5.708977					
2023-01-14 132.0299977929688 134.019998169453 131.6600004621093 133.759994683594 5.708977					
2023-01-18 136.82000732421875 138.61000061835358 135.02999877929688 135.2100067130872 6.9672087					
2023-01-19 134.8000018510547 136.1300007246094 135.27000427246094 135.27000427246094 5.820047					

Figure 28:L'affichage des résultats des requêtes SQL-partie2

9. Prédiction du score de sentiment des textes en utilisant Spark ML

Dans ce script PySpark, on utilise un modèle préalablement entraîné pour prédire les étiquettes de sentiment à partir d'une colonne de texte dans un DataFrame du New York Times. Les résultats sont ensuite filtrés et affichés, montrant les 7 premières lignes avec une colonne de texte tronquée, les scores de sentiment prédicts, et les étiquettes associées ("positive", "negative", "neutral").

```

spark
from pyspark.sql import PipelineModel
text_file_path = "hdfs://commodity:8000/user/test_model"
loaded_model = PipelineModel.load(hdfs_file_path)

# Spark
from pyspark.sql import SparkSession
from pyspark.ml import PipelineModel
from pyspark.ml.functions import col, udf, length, when
from pyspark.sql.types import StringType
# Assuming SparkSession is already created
# If not, create it using SparkSession.builder.appName("YourAppName").getOrCreate()
# Select the relevant column for prediction
text_column = "abstract"
ny_text_data = result_my_selection(text_column).alias("Sentence")
# Define a UDF to map sentiment labels
map_sentiment_label_udf = udf(lambda prediction: 'positive' if prediction == 1.0 else ('negative' if prediction == -1.0 else 'neutral'), StringType())
# Apply the loaded model to predict sentiment scores
ny_sentence_result = loaded_model.transform(ny_text_data)
# Filter out rows where "Sentence" is not null
ny_sentence_result = ny_sentence_result.filter(col("Sentence").isNotNull())
# Extract the first 7 words of each sentence
ny_sentence_result = ny_sentence_result.withColumn("truncated_sentence", when((length("Sentence") > 15) & (col("Sentence") != "")) , col("Sentence").substr(1, 15) + "...").otherwise(col("Sentence")))
# Select relevant columns for display
ny_sentence_result = ny_sentence_result.select("truncated_sentence", "prediction")
# Apply the UDF to create a new column with sentiment labels
ny_sentence_result = ny_sentence_result.withColumn("sentiment_label", map_sentiment_label_udf(col("Sentence")))
# Show the truncated results
ny_sentence_result.select("truncated_sentence", "Sentiment", "Sentiment_Label").limit(7).show(truncate=False)

```

Task 1 sec. Last updated by anonymous at December 21 2022, 8:04:42 PM.

Figure 29: Prédiction du score de sentiment des textes en utilisant Spark ML

```

lecture_data_spark_sql_dernier: x +
localhost:8082/#/notebook/2jJZN3C1

```

```

spark
from pyspark.sql import SparkSession
from pyspark.ml import PipelineModel
from pyspark.ml.functions import col, udf, length, when
from pyspark.sql.types import StringType
# Assuming SparkSession is already created
# If not, create it using SparkSession.builder.appName("YourAppName").getOrCreate()
# Select the relevant column for prediction
text_column = "abstract"
ny_text_data = result_my_selection(text_column).alias("Sentence")
# Define a UDF to map sentiment labels
map_sentiment_label_udf = udf(lambda prediction: 'positive' if prediction == 1.0 else ('negative' if prediction == -1.0 else 'neutral'), StringType())
# Apply the loaded model to predict sentiment scores
ny_sentence_result = loaded_model.transform(ny_text_data)
# Filter out rows where "Sentence" is not null
ny_sentence_result = ny_sentence_result.filter(col("Sentence").isNotNull())
# Extract the first 7 words of each sentence
ny_sentence_result = ny_sentence_result.withColumn("truncated_sentence", when((length("Sentence") > 15) & (col("Sentence") != "")) , col("Sentence").substr(1, 15) + "...").otherwise(col("Sentence")))
# Select relevant columns for display
ny_sentence_result = ny_sentence_result.select("truncated_sentence", "prediction")
# Apply the UDF to create a new column with sentiment labels
ny_sentence_result = ny_sentence_result.withColumn("sentiment_label", map_sentiment_label_udf(col("Sentence")))
# Show the truncated results
ny_sentence_result.select("truncated_sentence", "Sentiment", "Sentiment_Label").limit(7).show(truncate=False)

```

Task 21 sec. Last updated by anonymous at December 21 2022, 9:04:02 PM.

10. Ajout de nouvelles colonnes dans les données NYT

On analyse les données de sentiments du New York Times en tronquant les abstracts, calculant le nombre total de abstracts et la moyenne des sentiments par jour, fournissant un résumé global des statistiques par jour, et affichant les résultats triés par date.

```

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf, length, when, date_format, sum, format_number, count
from pyspark.sql.types import StringType

# Assuming SparkSession is already created
# If not, create it using SparkSession.builder.appname("yourappname").getOrCreate()

# Define a UDF to map sentiment labels
map_sentiment_label_udf = udf(lambda prediction: "positive" if prediction == 0.0 else ("negative" if prediction == -1.0 else "neutral"), StringType())

# Assuming you have a DataFrame with sentiment labels named result_my_sentiment
# Add a column for the length of each sentence
result_my_sentiment = result_my_sentiment.withColumn("truncated_sentence", when(length("sentence") > 187) & (col("sentence").substr(1, 7) != "...").otherwise(col("sentence")))

# Calculate the total number of sentences for each day
result_my_count_sentences = result_my_sentiment.groupby(date_format("pub_date", "yyyy-MM-dd").alias("pub_date"), "truncated_sentence").agg(count("*").alias("totalSentences"))

# Add truncated_sentence column to result_my_avg_sentence DataFrame
result_my_avg_sentence = result_my_count_sentences.groupby("pub_date", "truncated_sentence").agg(
    sum(when(col("sentiment_label") == 1, 1).otherwise(0)).alias("numPositive"),
    sum(when(col("sentiment_label") == 0, 1).otherwise(0)).alias("numNegative"),
    sum(when(col("sentiment_label") == -1, 1).otherwise(0)).alias("numNeutral")
)

# Join the average sentiment DataFrame with the total count DataFrame
result_my_final = result_my_avg_sentence.join(result_my_count_sentences, ["pub_date", "truncated_sentence"])
result_my_final = result_my_final.withColumn("avgSentiment", when(result_my_final["numPositive"] > 0, result_my_final["numPositive"] / result_my_final["totalSentences"]).otherwise(result_my_final["numNegative"] / result_my_final["totalSentences"]))

# Sort the Dataframe by date
result_my_summary = result_my_final.orderBy("pub_date")

# Show the results
result_my_summary.show(truncate=False)

```

pub_date	TotalSentences	numPositv	numNegative	numNeutral	AVGsentiment
2023-12-01 194	85	4	105	0.46	
2023-12-02 90	44	0	46	0.49	
2023-12-03 64	30	1	33	0.48	
2023-12-04 102	40	1	61	0.40	
2023-12-05 170	80	0	90	0.47	
2023-12-06 163	72	1	90	0.45	
2023-12-07 187	88	5	94	0.50	
2023-12-08 161	70	2	89	0.45	
2023-12-09 82	39	0	43	0.48	
2023-12-10 70	35	2	33	0.53	
2023-12-11 129	59	4	66	0.49	
2023-12-12 154	73	1	80	0.48	
2023-12-13 139	66	4	69	0.50	
2023-12-14 158	79	2	77	0.51	
2023-12-15 160	105	0	105	0.50	

Took 2 sec. Last updated by anonymous at December 21 2023, 5:14:15 PM.

```

# Add the provided data to result_nyt
# additional_data = spark.sql(""""
#     SELECT *
#     FROM nyt_data
#     """)

additional_data = spark.sql("""
    SELECT abstract,snippet,lead_paragraph,source,date_format(pub_date, 'yyyy-MM-dd') as pub_date,document_type,section_name,url
    FROM nyt_data
""")

# Show the results
# additional_data.show(truncate=False)

# Join based on "pub_date" only
result_nyt_final_with_additional_data = result_nyt_summary.join(
    additional_data.select("pub_date", "abstract", "snippet", "lead_paragraph", "source", "document_type", "section_name", "url"),
    "pub_date",
    "left_outer"
)

# Drop the results
result_nyt_final_with_additional_data.show(truncate=True)
hdfs_path_csv = "hdfs://namenode:9000/user/mnDataFrame.csv"
# hdfs_path_csv = "/opt/zeppelin/mnDataFrame2.csv"
# save Dataframe to HDFS in CSV format
result_nyt_final_with_additional_data_grouped.write.csv(hdfs_path_csv, mode="overwritte", header=True)

```

pub_date	TotalSentences	numPositive	numNegative	numNeutral	AVGsentiment	abstract	snippet	lead_paragraph	source	document_type	section_name	url
2023-12-01	194	85	41	188	0.46	The Heritage Foun...[The Heritage Foun... [An influential co...[The New York Times]	article	U.S. nyt://article/919...				
2023-12-01	194	85	41	188	0.46	The Heritage Foun...[The Heritage Foun... [The music mogul ...[The New York Times]	multimedia	U.S. nyt://interactive...				
2023-12-01	194	85	41	188	0.46	The music mogul ... [A few months ago,...[The New York Times]	article	Arts nyt://article/4d1...				
2023-12-01	194	85	41	188	0.46	Wardlow, aka...[Wardlow, aka... [A few months ago,...[The New York Times]	article	U.S. nyt://article/111...				
2023-12-01	194	85	41	188	0.46	Going without me...[Going without me... [Hurricane...[The New York Times]	article	World nyt://article/987...				
2023-12-01	194	85	41	188	0.46	Also, George Sant... [Also, George Sant... [A weeklong cease...[The New York Times]	article	Briefing nyt://article/25f...				
2023-12-01	194	85	41	188	0.46	Hijacking out Hemas ... [Hijacking out Hemas ... [As Iranian debates...[The New York Times]	article	Opinion nyt://article/eb...				
2023-12-01	194	85	41	188	0.46	The New York Tim... [The New York Tim... [A weeklong cease...[The New York Times]	multimedia	Op-Ed nyt://video/349...				
2023-12-01	194	85	41	188	0.46	Justice Dept. Contra... [Justice Dept. Contra... [A weeklong cease...[The New York Times]	multimedia	Op-Ed nyt://video/349...				
2023-12-01	194	85	41	188	0.46	Can you sort 8 hi... [Can you sort 8 hi... [Can you sort 8 hi... [The New York Times]	multimedia	The Upshot nyt://interactive...				
2023-12-01	194	85	41	188	0.46	House Republicans... [House Republicans... [House Republicans... [The New York Times]	article	U.S. nyt://article/c97...				
2023-12-01	194	85	41	188	0.46	Israel has releas... [Israel has releas... [Israel has releas... [The New York Times]	article	World nyt://article/1d6...				
2023-12-01	194	85	41	188	0.46	The police are an... [The police are an... [The police are an... [The Los Angeles p...[The New York Times]	article	U.S. nyt://article/441...				
2023-12-01	194	85	41	188	0.46	Meet Progressive... [Meet Progressive... [Meet Progressive... [The New York Times]	article	New York nyt://article/469...				
2023-12-01	194	85	41	188	0.46	Two researchers w... [Two researchers w... [Two researchers w... [Five years ago,...[The New York Times]	article	Health nyt://article/454...				
2023-12-01	194	85	41	188	0.46	Gov. Greg Abbott'... [Gov. Greg Abbott'... [Gov. Greg Abbott'... [A federal appeals...[The New York Times]	article	U.S. nyt://article/3e6...				
2023-12-01	194	85	41	188	0.46	300 volunteers ... [300 volunteers ... [300 volunteers ... [300 volunteers ...[The New York Times]	article	U.S. nyt://article/974...				
2023-12-01	194	85	41	188	0.46	Alexander Chapel... [Alexander Chapel... [Alexander Chapel... [Less than a week...[The New York Times]	article	Health nyt://article/3e6...				
2023-12-01	194	85	41	188	0.46	Plus: fringe acc... [Plus: fringe acc... [Plus: fringe acc... [Maximilian Davis,...[The New York Times]	article	T Magazine nyt://article/948...				
2023-12-01	194	85	41	188	0.46	Maximilian Davis,... [Maximilian Davis,... [Maximilian Davis,... [Less than a week...[The New York Times]	article	T Magazine nyt://article/398...				

only showing top 20 rows

Figure 30:L'ajout de nouvelles colonnes dans les données NYT-partie1

On ajoute les colonnes suivantes dans les données New York Times :

- Une colonne nommée AVGsentiment dans laquelle vous mettez la moyenne des sentiments des textes publiés dans cette journée
- On ajoute une autre colonne appelé numPositv dans laquelle vous mettez le nombre de textes positives et une autre appelée numNegative pour les textes négatifs

lecture_data_spark_sql_derniere_version++

```
Zeppelin Notebook - Job
lecture_data_spark_sql_derniere_version++ default SPARK JOB FINISHED
```

```
sparkSQL
from pyspark.sql.functions import first
# Group by pub_date and select the first value for each group
result_my_final_with_additional_data_grouped = result_my_final_with_additional_data.groupby("pub_date").agg(
    first("numPositive").alias("numPositive"),
    first("numNegative").alias("numNegative"),
    first("numNeutral").alias("numNeutral"),
    first("numAdvertisement").alias("numAdvertisement"),
    first("abstract").alias("abstract"),
    first("snippet").alias("snippet"),
    first("lead_paragraph").alias("lead_paragraph"),
    first("source").alias("source"),
    first("document_type").alias("document_type"),
    first("section_name").alias("section_name"),
    first("url").alias("url")
)

result_my_final_with_additional_data_grouped.show(truncate=True)

# Specify the HDFS path where you want to save the CSV file
hdfs_path_csv = "hdfs://namenode:9000/user/result.csv"

# Save DataFrame to HDFS in CSV Format
result_my_final_with_additional_data_grouped.write.csv(hdfs_path_csv, mode="overwrite", header=True)
```

Figure 31:L'ajout de nouvelles colonnes dans les données NYT-partie2

11. Enregistrement du nouveau DataFrame obtenu dans apache Druid

```
# Specify the HDFS path where you want to save the CSV file
hdfs_path_csv = "hdfs://namenode:9000/user/result.csv"

# Save DataFrame to HDFS in CSV format
result_nyt_final_with_additional_data_grouped.write.csv(hdfs_path_csv, mode="overwrite", header=True)
```

Voilà, les étapes à suivre pour enregistrer le nouveau dataframe dans une nouvelle datastorer Druid :

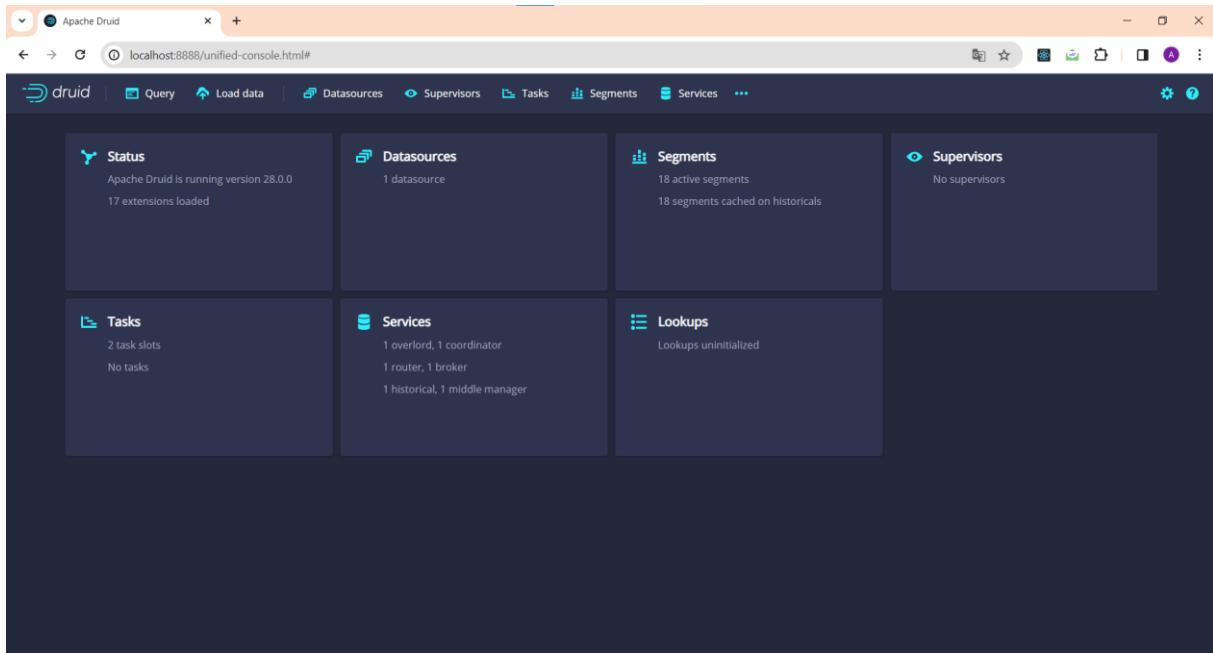


Figure 32:L'interface utilisateur de Apache Druid

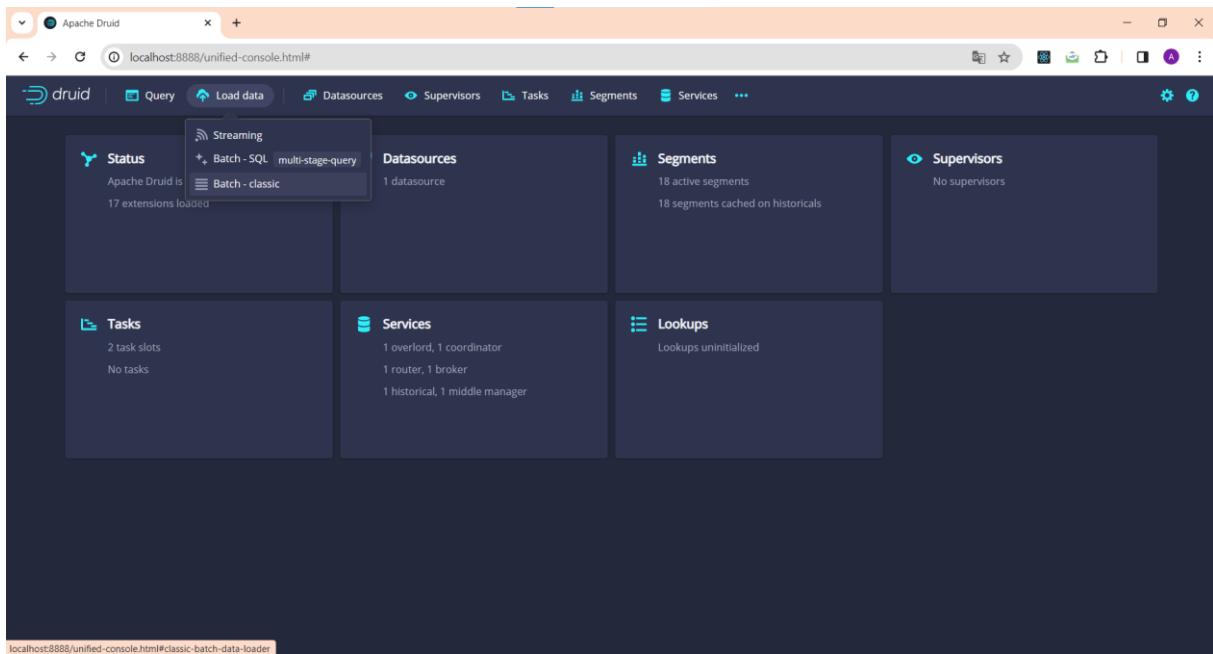


Figure 33:Le choix de Batch-classic

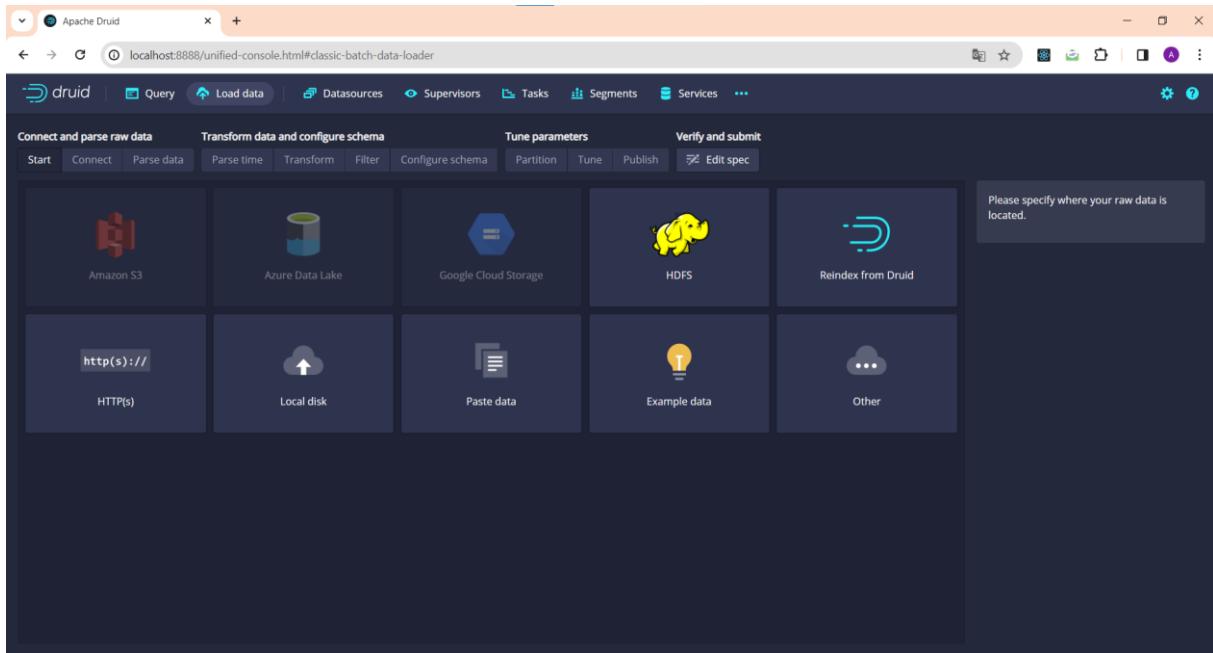


Figure 34: cliquer sur HDFS

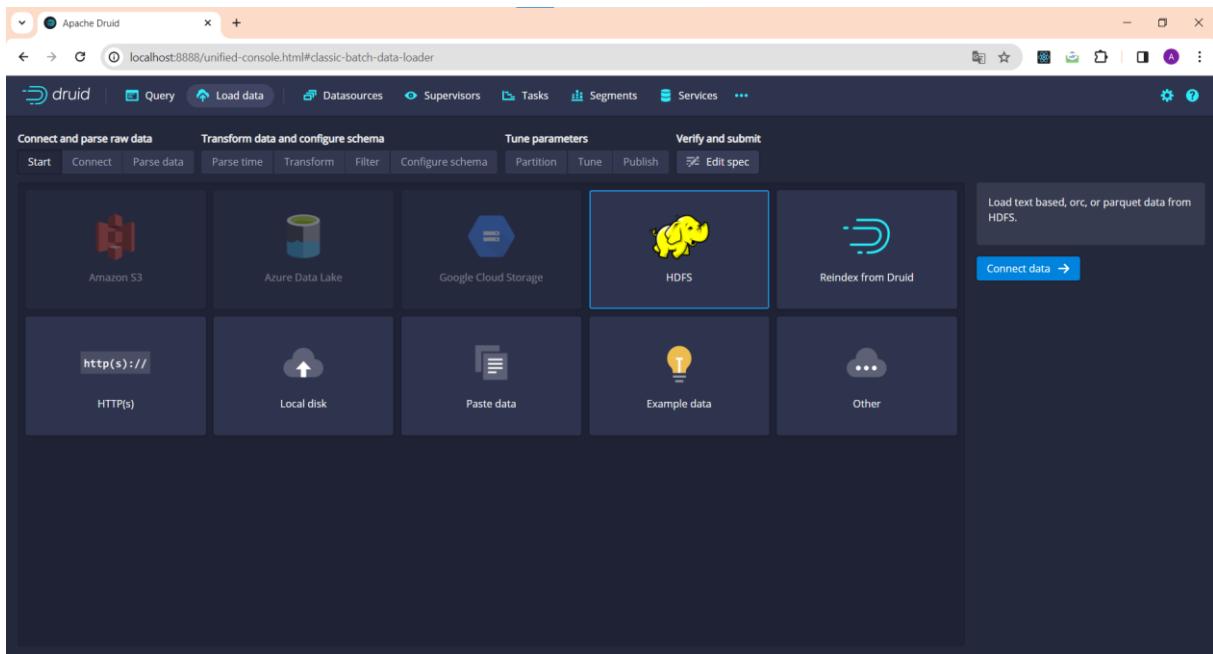


Figure 35: Cliquer sur Connect data

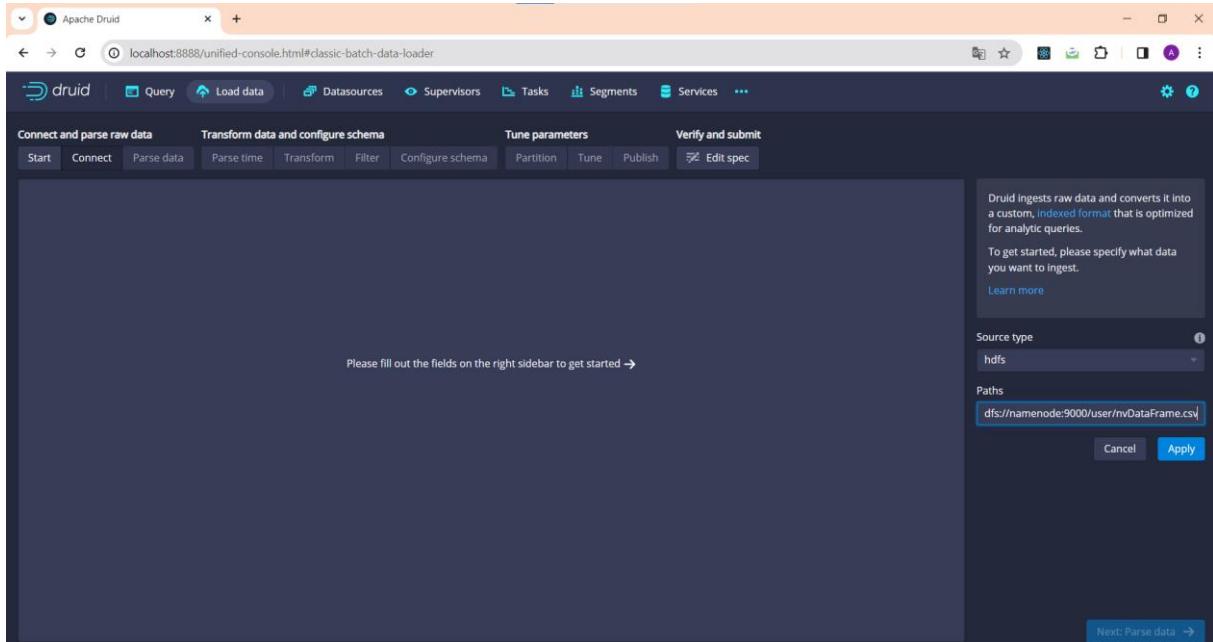
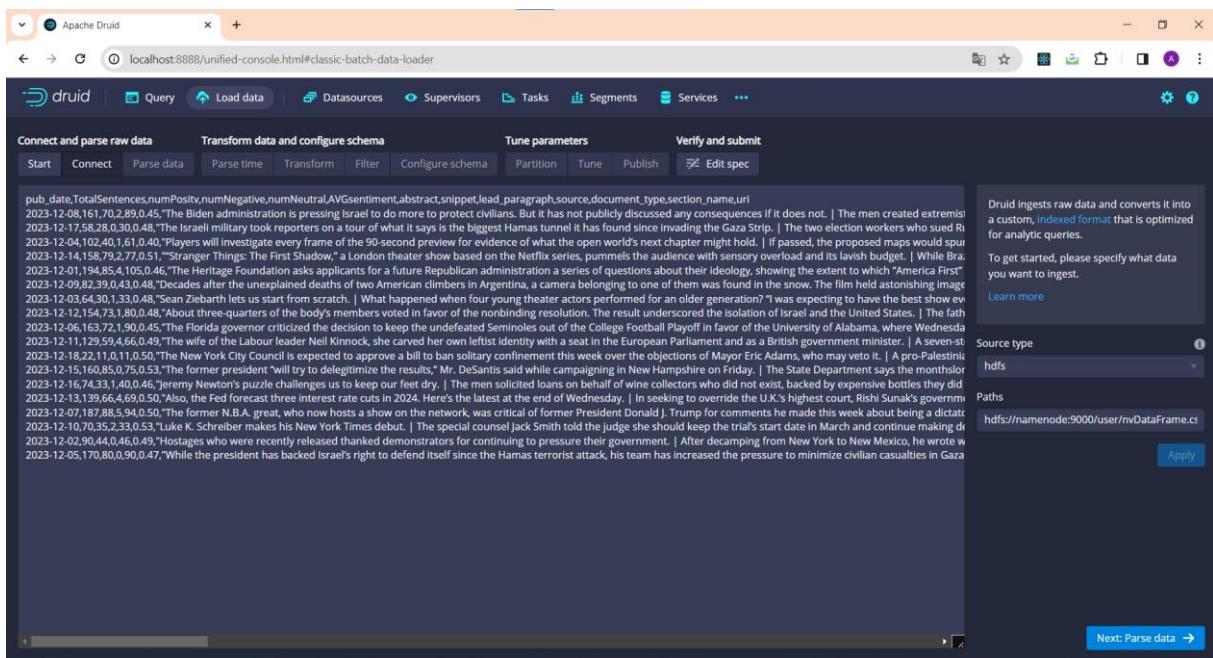


Figure 36:L'insertion du chemin hdfs de nouveau dataframe



Apache Druid

localhost:8888/unified-console.html#classic-batch-data-loader

druid query Load data Datasources Supervisors Tasks Segments Services ...

Connect and parse raw data Transform data and configure schema Tune parameters Verify and submit

Start Connect Parse data Parse time Transform Filter Configure schema Partition Tune Publish Edit spec

Search columns

pub_date	TotalSentences	numPosit	numNegative	numNeutral	AVGsentiment	abstract	snippet	lead_p
2023-12-08	161	70	2	89	0.45	The Biden admin...***	The Biden admin...***	The Bi
2023-12-17	58	28	0	30	0.48	The Israeli milita...***	The Israeli milita...***	The tui
2023-12-04	102	40	1	61	0.40	Players will invest...***	Players will invest...***	There v
2023-12-14	158	79	2	77	0.51	"Stranger Things:...***	"Stranger Things:...***	As the
2023-12-01	194	85	4	105	0.46	The Heritage Fou...***	The Heritage Fou...***	An infl
2023-12-09	82	39	0	43	0.48	Decades after the...***	Decades after the...***	Decade
2023-12-03	64	30	1	33	0.48	Sean Ziebarth let...***	Sean Ziebarth let...***	Jump t
2023-12-12	154	73	1	80	0.48	About three-quar...***	About three-quar...***	The U.I.
2023-12-06	163	72	1	90	0.45	The Florida gover...***	The Florida gover...***	An unc
2023-12-11	129	59	4	66	0.49	The wife of the La...***	The wife of the La...***	Glenys
2023-12-18	22	11	0	11	0.50	The New York Cit...***	The New York Cit...***	The Cit

Druid needs to parse data as columns. Determine the format of your data and ensure that the columns are accurately parsed.
If you have nested data, you can ingest it as json dimensions.
[Learn more](#)

Input format: csv
Skip header rows: 0
Find columns from header: False True
List delimiter: \u0001
Apply

Next: Parse time →

Apache Druid

localhost:8888/unified-console.html#classic-batch-data-loader

druid query Load data Datasources Supervisors Tasks Segments Services ...

Connect and parse raw data Transform data and configure schema Tune parameters Verify and submit

Start Connect Parse data Parse time Transform Filter Configure schema Partition Tune Publish Edit spec

Publish configuration

Datasource name: nouveau_DataFrame

Append to existing: True

Allow concurrent replace tasks (experimental): True

Parse error reporting

Log parse exceptions: True

Max parse exceptions: (unlimited)

Max saved parse exceptions: 0

Configure behavior of indexed data once it reaches Druid.

Next: Edit spec →

The screenshot shows the Apache Druid unified console interface. The URL is `localhost:8888/unified-console.html#classic-batch-data-loader`. The main area displays a JSON configuration for a new DataFrame:

```
{
  "type": "index_parallel",
  "spec": {
    "ioConfig": {
      "type": "index_parallel",
      "inputSource": {
        "type": "paths",
        "paths": "hdfs://namenode:9000/user/nvDataFrame.csv"
      },
      "inputFormat": {
        "type": "csv",
        "findColumnsFromHeader": true
      }
    },
    "tuningConfig": {
      "type": "index_parallel",
      "partitionsSpec": {
        "type": "dynamic"
      }
    },
    "dataschema": {
      "timestampSpec": {
        "column": "pub_date",
        "format": "auto"
      },
      "dimensionsSpec": {
        "dimensions": [
          {
            "type": "long",
            "name": "totalSentences"
          },
          {
            "type": "long",
            "name": "numPositive"
          },
          {
            "type": "long",
            "name": "numNegative"
          },
          {
            "type": "long",
            "name": "numNeutral"
          }
        ]
      }
    }
  }
}
```

A summary message on the right states: "Druid begins ingesting data once you submit a JSON ingestion spec. If you modify any values in this view, the values entered in previous sections will update accordingly. If you modify any values in previous sections, this spec will automatically update." Below it, a button says "Submit" with a file icon.

Figure 37:Le schéma de nouveau dataframe

The screenshot shows the Apache Druid unified console tasks page. The URL is `localhost:8888/unified-console.html#tasks/group_id=index_parallel_nouveau_DataFrame_pdf0fpfj_2023-12-23T12:32:42.078Z`. The tasks table has the following data:

Task ID	Group ID	Type	Datasource	Status	Created time	Duration	Locality
index_parallel_nouveau_DataFrame_p		index_parallel	nouveau_DataFrame	PENDING	2023-12-23T12:32:42.181Z	0:00:01	

Figure 38:L'ajout de nouveau dataframe est entrain de s'exécuter

Figure 39:L'ajout de nouveau dataframe est entrain est exécuté avec succès

Figure 40:La création de nouveau datasource druid "nouveau_dataframe"

12. Visualisation du résultat de traitement effectué en utilisant Streamlit

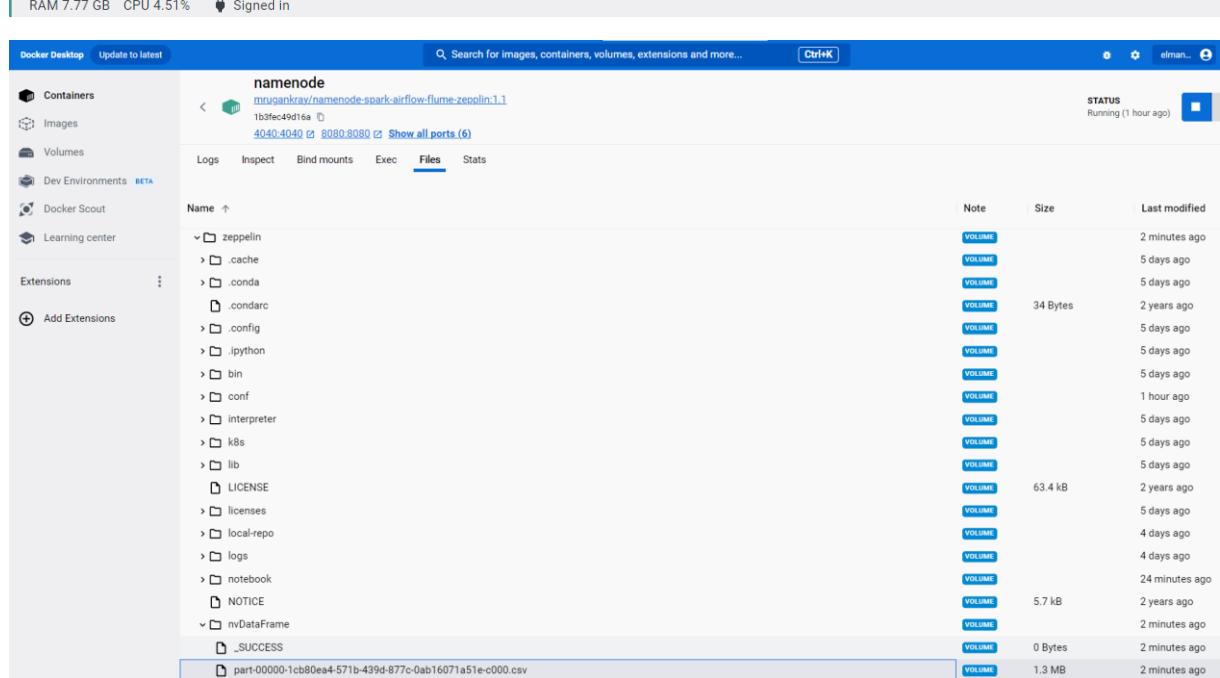
Téléchargement du fichier csv de nouveau dataframe :

```
C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master>curl -o "C:/Users/EliteBook G4/Desktop/Mini-Projet-BigData/materials/mise-en-place-d-un-data-lake-master/nvDataFrame.csv" "http://localhost:9000/user/nvDataFrame.csv?op=OPEN"
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          0     0      0      0      0      0      0      0 --:--:-- --:--:-- --:--:-- 1824
C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master>
```

⇒ curl -o "C:/Users/EliteBook G4/Desktop/Mini-Projet-BigData/materials/mise-en-place-d-un-data-lake-master/nvDataFrame.csv"
<http://localhost:9000/user/nvDataFrame.csv?op=OPEN>

Ou bien il suffit de taper la commande suivante :

```
# hadoop fs -copyToLocal hdfs://namenode:9000/user/nvDataFrame.csv "/opt/zeppelin/nvDataFrame"
2023-12-21 17:25:16,534 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
#
```



Docker Desktop Update to latest

namenode

namenode

1b3fec49d1ea 4040:4040 8080:8080 Show all ports (6)

Status: Running (1 hour ago)

Logs Inspect Bind mounts Exec Files Stats

Name: zeppelin

File Tree:

- zeppelin
- .cache
- .conda
- .condarc
- .config
- .ipython
- bin
- conf
- interpreter
- k8s
- lib
- LICENSE
- licenses
- local-repo
- logs
- notebook
- NOTICE
- nvDataFrame
- _SUCCESS
- part-00000-1cb80ea4-571b-439d-877c-0ab16071a51e-c000.csv

Note Size Last modified

Note	Size	Last modified
VOLUME	34 Bytes	2 minutes ago
VOLUME		5 days ago
VOLUME		5 days ago
VOLUME	63.4 kB	2 years ago
VOLUME		5 days ago
VOLUME		5 days ago
VOLUME		1 hour ago
VOLUME		5 days ago
VOLUME	5.7 kB	2 years ago
VOLUME	0 Bytes	2 minutes ago
VOLUME	1.3 MB	2 minutes ago

zeppelin		VOLUME	2 minutes ago	drwxrwxr-x
.cache		VOLUME	5 days ago	drwxr-xr-x
.conda		VOLUME	5 days ago	drwxr-xr-x
.condarc		VOLUME	2 years ago	-rw-r--r--
.config		VOLUME	5 days ago	drwxr-xr-x
.ipython		VOLUME	5 days ago	drwxr-xr-x
bin		VOLUME	5 days ago	drwxr-xr-x
conf		VOLUME	1 hour ago	drwxrwxr-x
interpreter		VOLUME	5 days ago	drwxr-xr-x
k8s		VOLUME	5 days ago	drwxr-xr-x
lib		VOLUME	5 days ago	drwxr-xr-x
LICENSE		VOLUME	63.4 kB	-rw-r--r--
licenses		VOLUME	5 days ago	drwxr-xr-x
local-repo		VOLUME	4 days ago	drwxr-xr-x
logs		VOLUME	4 days ago	drwxrwxr-x
notebook		VOLUME	24 minutes ago	drwxrwxr-x
NOTICE		VOLUME	5.7 kB	-rw-r--r--
nvDataFrame		VOLUME	2 years ago	2 minutes ago
_SUCCESS		VOLUME	0 Bytes	drwxr-xr-x
part-00000-1cb80ea4-571b-439d-877c-0ab16071a51e-c000.csv		VOLUME	1.3 MB	-rw-r--r--

Puis vous pouvez télécharger le fichier qui se trouve dans le chemin **/opt/zeppelin/nvDataFrame** puis importer le dans votre dossier de travail ,afin que vous puissiez l'utiliser par la suite dans la visualisation avec Streamlit.

```
Stopping...
PS C:\Users\EliteBook G4\Desktop\Mini-Projet-BigData\materials\mise-en-place-d-un-data-lake-master> streamlit run streamlit_app.py
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.131:8501
```

Tableau de données :

- Affiche l'ensemble du jeu de données sous forme de tableau, montrant différentes colonnes telles que la date de publication, la moyenne de sentiments de textes publiés pour chaque jour, le nombre de sentiments positives pour chaque jour et autres.



News Sentiment Analysis

Table of Data

	pub_date	TotalSentences	numPositive	numNegative	numNeutral	AVGsentiment	abstract
8	2023-12-06	163	72	1	90	0.45	The Florida gove
9	2023-12-11	129	59	4	66	0.49	The wife of the L
10	2023-12-18	22	11	0	11	0.5	The New York Cit
11	2023-12-15	160	85	0	75	0.53	The former presi
12	2023-12-16	74	33	1	40	0.46	Jeremy Newton'
13	2023-12-13	139	66	4	69	0.5	Also, the Fed for
14	2023-12-07	187	88	5	94	0.5	The former N.B.I
15	2023-12-10	70	35	2	33	0.53	Luke K. Schreibe
16	2023-12-02	90	44	0	46	0.49	Hostages who wr
17	2023-12-05	170	80	0	90	0.47	While the presid

Figure 41:Le tableau des données du nouveau_dataframe

Nombre de sentiments positifs par jour :

- Présente un graphique à barres illustrant le nombre total de sentiments positifs agrégés par jour.



Number of Positive Sentiments per Day

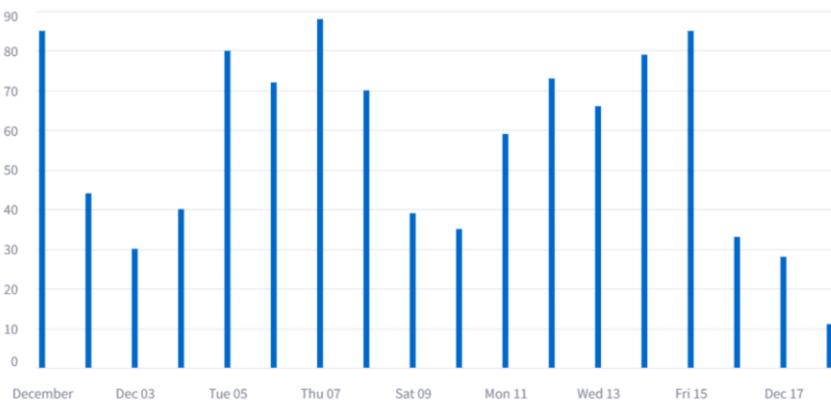


Figure 42:Le nombre de sentiment positifs par jour

Nombre de sentiments négatifs par jour :

- Fournit un graphique à barres montrant le nombre total de sentiments négatifs agrégés par jour.

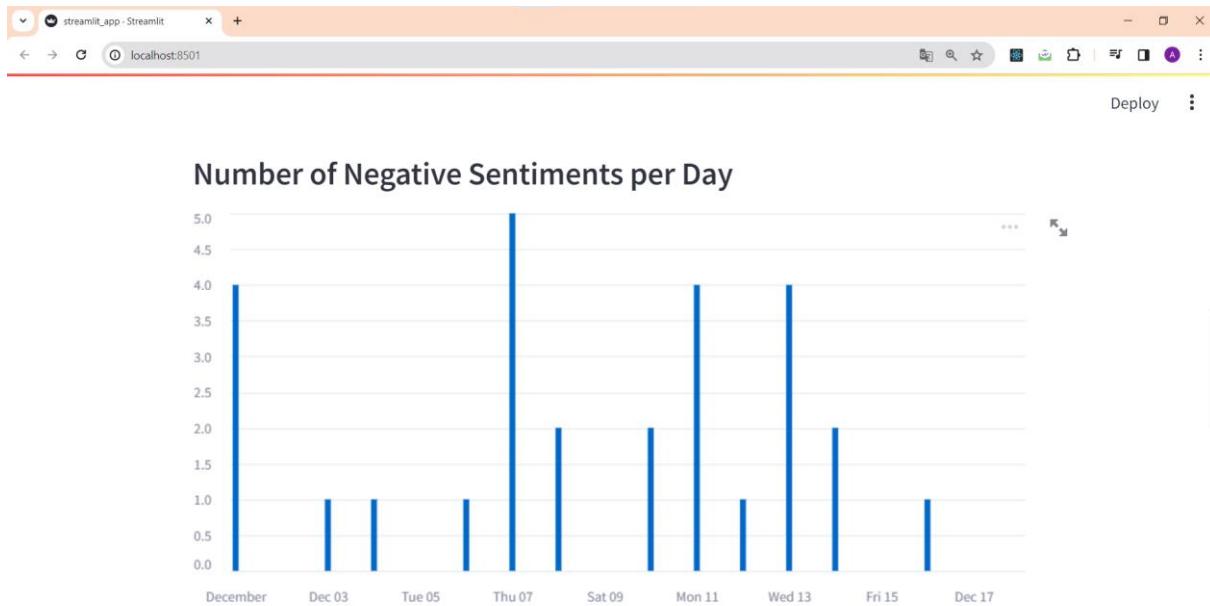


Figure 43:Le nombre de sentiment négatifs par jour

Sentiment moyen par jour :

- Affiche un graphique linéaire représentant le score de sentiment moyen par jour.

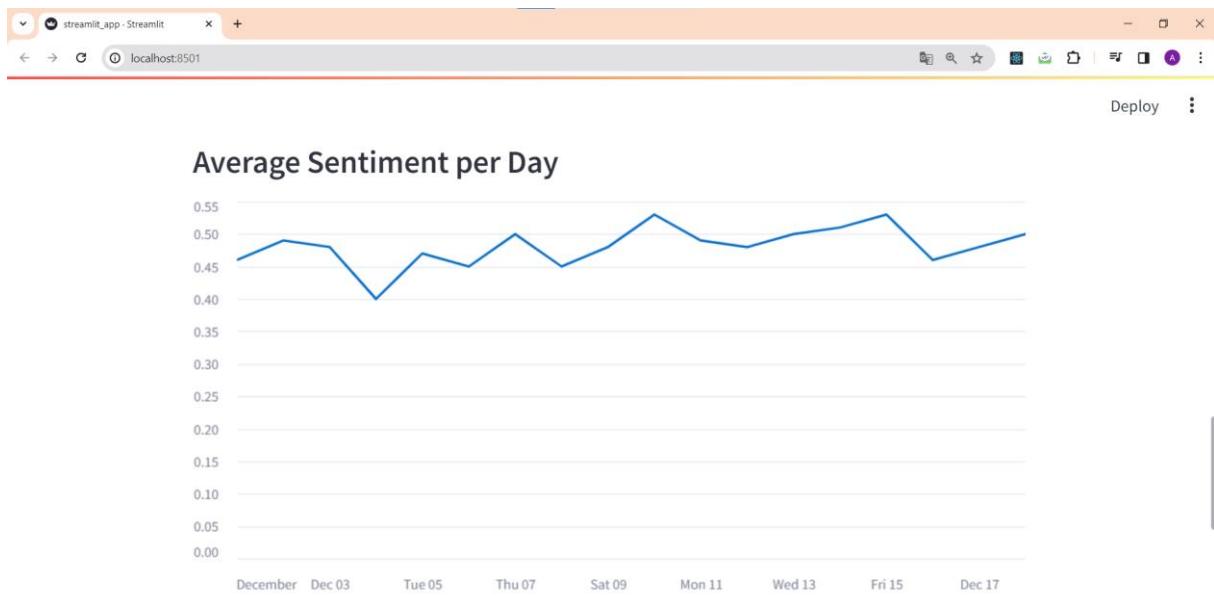


Figure 44:La moyenne des sentiments par jour

Nombre d'articles par jour :

- Affiche un graphique à barres représentant le nombre total d'articles par jour.

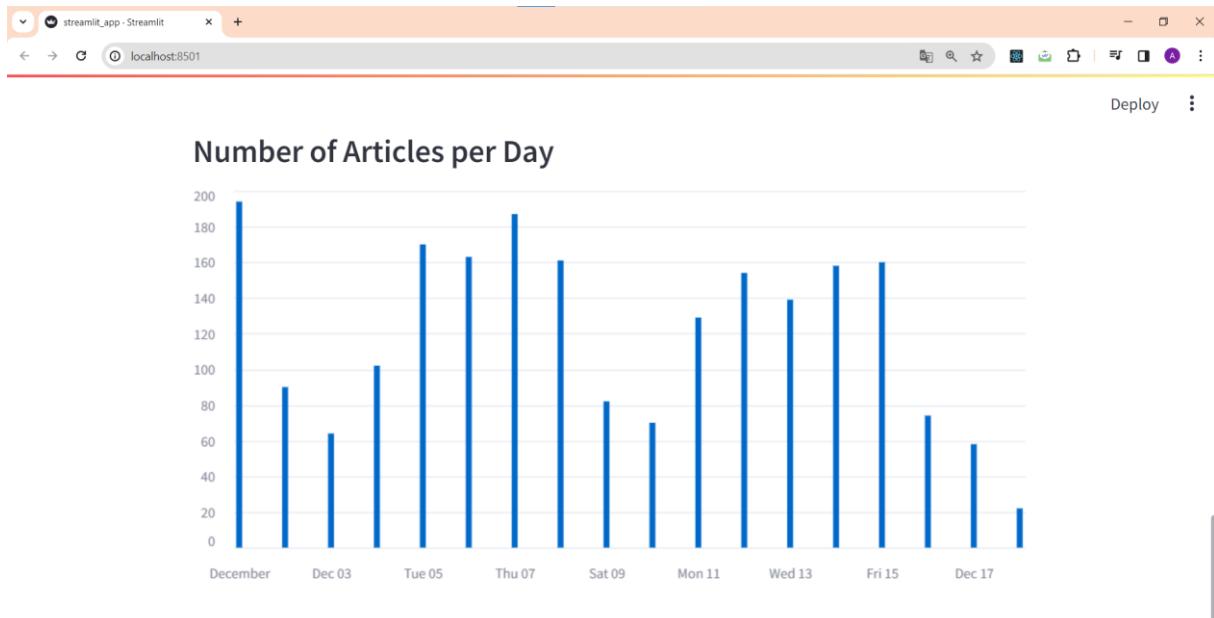


Figure 45:Le nombre des articles par jour

Distribution des scores de sentiment :

- Présente un histogramme affichant la distribution des scores de sentiment moyen à travers tous les points de données.

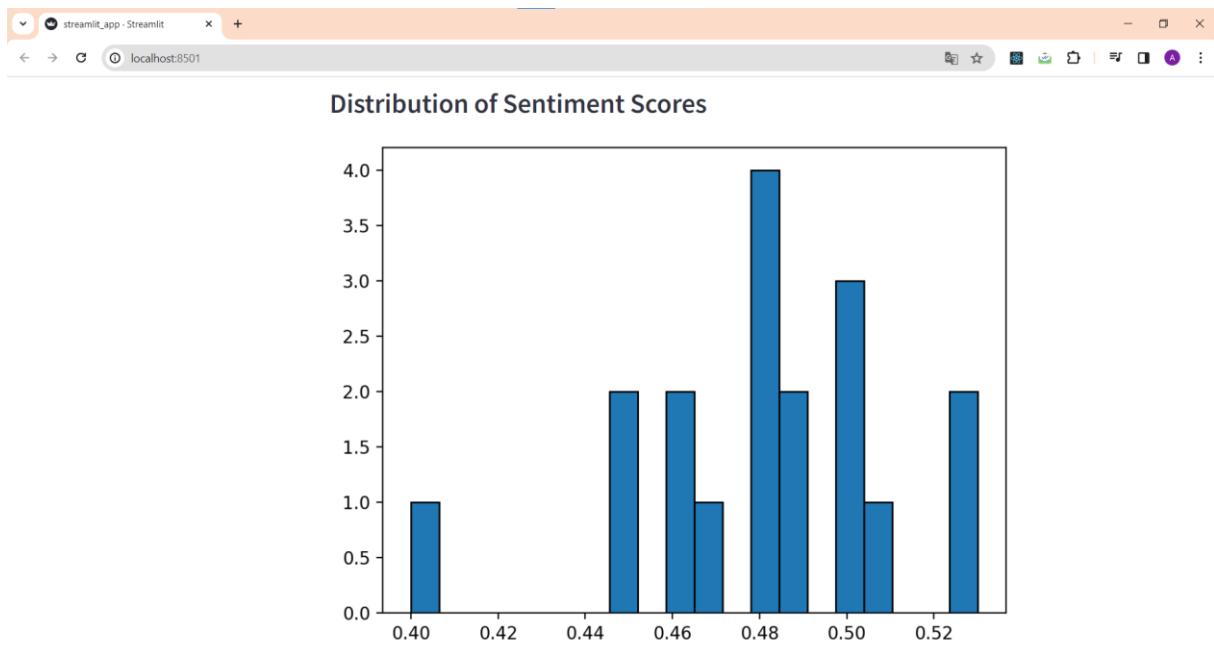


Figure 46: Répartition des scores de sentiments

Comptage des occurrences de sections :

- Affiche un tableau présentant le nombre d'occurrences pour chaque section prédéfinie (par exemple, U.S., World, Arts) dans les données

	Section	Occurrences
0	U.S.	18
1	New York	18
2	Crosswords & Games	18
3	Climate	16
4	World	18
5	Arts	18
6	The Learning Network	14
7	Briefing	18
8	Business Day	17
9	Corrections	18

Figure 47: Le nombre d'occurrences pour chaque section

Diagramme en barres du nombre d'occurrences :

- Fournit un diagramme en barres avec des couleurs différentes pour chaque section, illustrant le nombre d'occurrences pour chaque section.

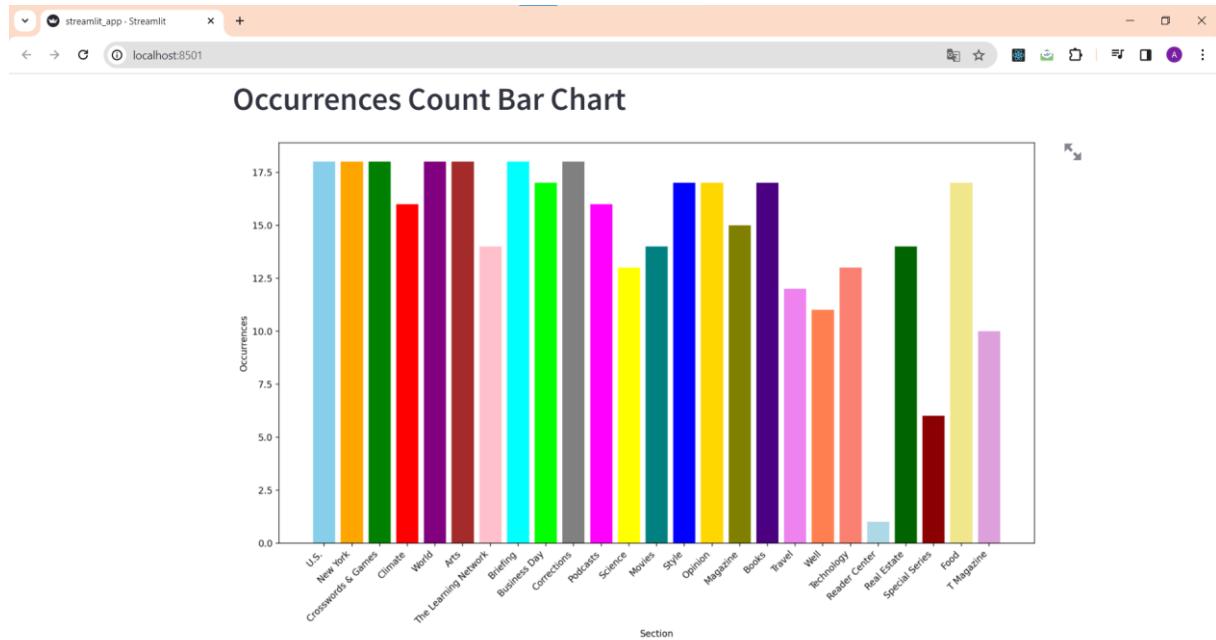


Figure 48: Le graphe à barres pour calculer le nombre d'occurrences pour chaque section

Nuage de mots des résumés(abstracts) :

- Génère un nuage de mots basé sur les résumés(abstracts) des articles, où la taille de chaque mot représente sa fréquence dans le texte.



Figure 49:Le résumé de nuage de mots

