

Projet : Pipeline Airflow pour l'Analyse des Données YouTube

EL Assioui Ayoub

December 18, 2024

Contents

1	Introduction	2
2	Structure du Projet	3
3	Description des Étapes du Pipeline	4
3.1	Scraping des Données	4
3.2	Préprocessing des Données	4
3.3	Analyse des Données avec HuggingFace	4
3.4	Visualisation des Résultats	4
3.5	Conteneurisation avec Docker	5
3.6	Résultats:	5
4	Conclusion	8

Chapter 1

Introduction

Ce rapport présente un pipeline automatisé pour l'extraction, le traitement, l'analyse et la visualisation des données issues de la plateforme YouTube. Le pipeline utilise Apache Airflow pour orchestrer les différentes étapes, et HuggingFace pour l'analyse des sentiments des données. L'intégration d'un conteneur Docker permet une exécution simplifiée du projet.

Les objectifs principaux du projet sont :

- Extraire des données YouTube (scraping).
- Nettoyer et prétraiter les données.
- Analyser les titres de vidéos avec un modèle HuggingFace.
- Visualiser les résultats sous forme de graphiques.

Chapter 2

Structure du Projet

Le projet est structuré de manière modulaire pour assurer une bonne organisation et évolutivité. Voici les principales composantes :

- **DAGs Airflow** : Chaque étape du pipeline est implémentée dans un DAG distinct.
- **Données** : Les données sont sauvegardées localement avec un horodatage pour assurer leur traçabilité.
- **Analyse HuggingFace** : Utilisation d'un modèle pré-entraîné pour réaliser l'analyse des sentiments.
- **Visualisation** : Génération de graphiques pour interpréter les résultats.
- **Conteneurisation Docker** : Automatisation de l'exécution du pipeline dans un environnement conteneurisé.

Chapter 3

Description des Étapes du Pipeline

3.1 Scraping des Données

L'étape initiale consiste à extraire les données depuis l'API YouTube. Les vidéos d'une chaîne YouTube donnée sont collectées, et leurs informations (titre, ID de vidéo, date de publication) sont sauvegardées dans des fichiers CSV localement. Chaque fichier est horodaté pour assurer une bonne organisation.

3.2 Préprocessing des Données

Les données collectées sont nettoyées et transformées. Les étapes incluent :

- Conversion des dates en format approprié.
- Calcul de la longueur des titres des vidéos.
- Sauvegarde des données prétraitées dans un fichier CSV.

3.3 Analyse des Données avec HuggingFace

Pour analyser les titres des vidéos, un modèle de classification des sentiments de HuggingFace est utilisé. L'analyse permet de déterminer si un titre est perçu comme *positif*, *négatif* ou *neutre*. Les résultats sont ajoutés aux données prétraitées.

3.4 Visualisation des Résultats

Une étape de visualisation est ajoutée pour représenter graphiquement les résultats. Les visualisations comprennent :

- Une répartition des sentiments des titres de vidéos.
- Une comparaison de la longueur des titres en fonction des sentiments.

Les graphiques sont sauvegardés localement pour consultation.

3.5 Conteneurisation avec Docker

Pour faciliter l'exécution du pipeline, un fichier Dockerfile a été créé. Ce fichier permet de construire un conteneur contenant Airflow, ainsi que toutes les dépendances nécessaires pour l'extraction, le traitement et l'analyse des données.

Le conteneurisation offre plusieurs avantages :

- Simplification du déploiement.
- Portabilité entre environnements.
- Reproductibilité du pipeline.

3.6 Résultats:

- Création du conteneur à l'aide de docker

```
docker build -t airflow-youtube .
```

```
[+] Building 1085.0s (8/10)                                docker:desktop-linux
=> => transferring context: 2B                               0.0s
=> [1/6] FROM docker.io/apache/airflow:2.7.0-python3.9@sha256:927139b7cffe8eb719f0b7c8cc3 0.1s
=> => resolve docker.io/apache/airflow:2.7.0-python3.9@sha256:927139b7cffe8eb719f0b7c8cc3 0.1s
=> [internal] load build context                             0.0s
=> => transferring context: 6.12kB                           0.0s
=> CACHED [2/6] WORKDIR /app                                0.0s
=> [3/6] COPY ./dags /opt/airflow/dags                     0.2s
=> [4/6] COPY ./requirements.txt /app/requirements.txt     0.1s
=> [5/6] RUN pip install --no-cache-dir -r /app/requirements.txt 1081.3s
```


- Création d'un compte à Airflow

```
TypeError: module object is not callable
(devoir3env) assioui@assioui-Latitude-5500:~/Devoir-part2$ docker exec -it 3369cf73ec5e68d9aebc45259267a15339dba9750f222fb26662d3c3200df4e9 bash
airflow@3369cf73ec5e:/app$ airflow users create \
  --username AyoubELAssioui \
  --firstname Admin \
  --lastname User \
  --role Admin \
  --email assioui69@gmail.com \
  --password A1t00nT4n
```

- Lancement du conteneur

```
docker run -p 8080:8080 airflow-youtube
```

- Mangement de dags avec airflow


DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
09:56 UTC
AU

Do not use **SQLite** as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. [Click here](#) for more information.

Do not use the **SequentialExecutor** in production. [Click here](#) for more information.

DAGs

All **1**
Active **1**
Paused **0**
Running **2**
Failed **0**


☒ Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
<input checked="" type="checkbox"/> youtube_scraping_dag airflow 2 @daily 1				2024-12-18, 09:55:16	2024-12-18, 00:00:00	6 ?

«
<
1
>
»

Showing 1-1 of 1 DAGs

Version: v2.7.0
Git Version: .release:c08c82e9dd0e4aaba5121519819a636d635210


DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
09:53 UTC
AU

12 / 18 / 2024 , 09 : 53 : 38 AM
25
All Run Types
All Run States
Clear Filters
Auto-refresh

Press **shift** + **/** for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status

DAG
youtube_scraping_dag
Details
Graph
Gantt
Code

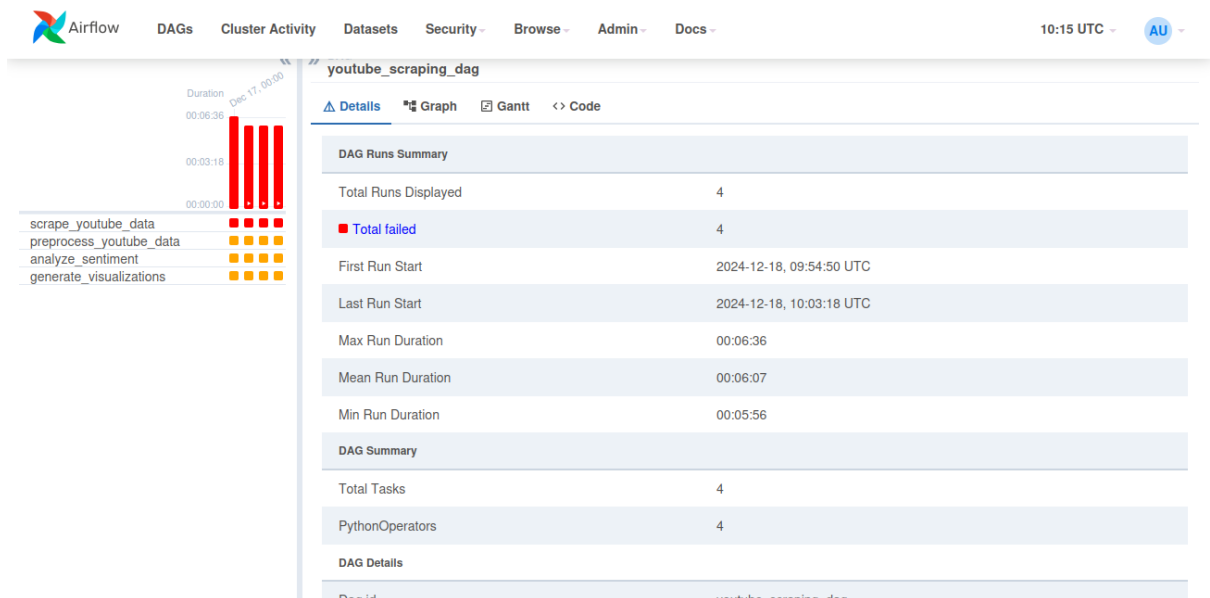
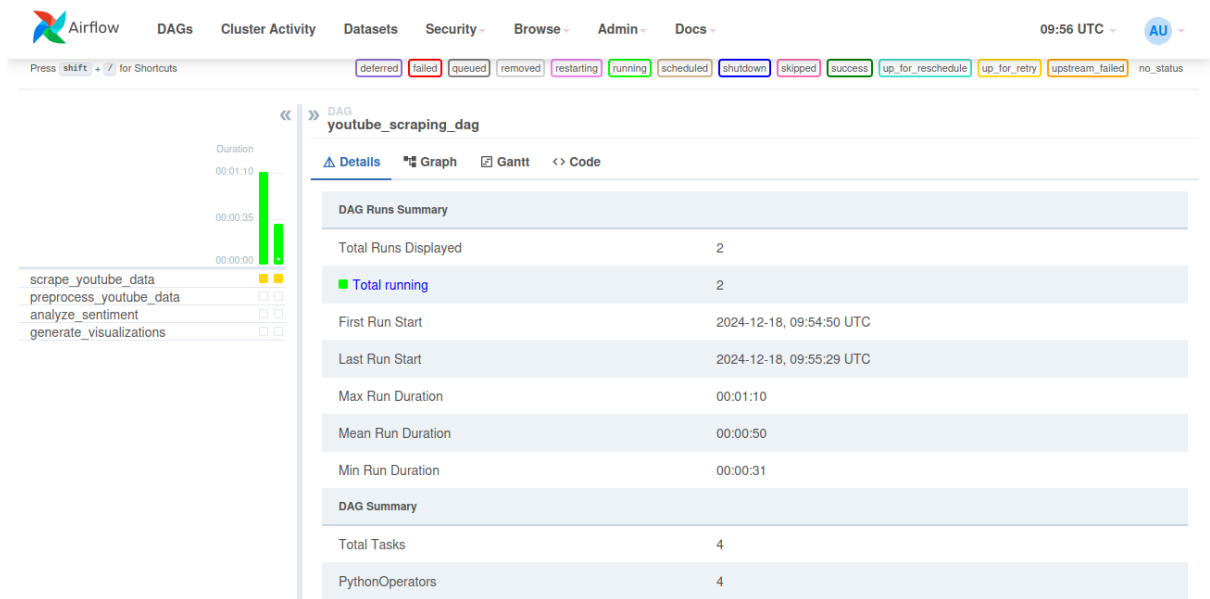
Layout:
Left -> Right

```

graph LR
    A[scrape_youtube_data  
PythonOperator] --> B[preprocess_youtube_data  
PythonOperator]
    B --> C[analyze_sentiment  
PythonOperator]
    C --> D[generate_visualizations  
PythonOperator]

```

scrape_youtube_data
preprocess_youtube_data
analyze_sentiment
generate_visualizations



Chapter 4

Conclusion

Ce projet démontre une approche modulaire et automatisée pour l'analyse des données YouTube. L'utilisation d'Apache Airflow permet d'orchestrer chaque étape, tandis que HuggingFace offre une solution efficace pour l'analyse des sentiments. La visualisation des résultats apporte des insights exploitables, et la conteneurisation avec Docker simplifie le déploiement. Jusqu'à la fin du délai pour ce projet il y a encore des erreurs dans ces dags. Si le problème est résolu le code sera adapté en github