Reporte de Proyecto Kafka

Estudiante: Inti Luna Avilés

Programa: Ingeniería de datos.
DATAHACK

2024-06-06

Resumen

Se ha creado una *data pipeline* para procesar datos de opiniones de un producto (maletas) utilizando kafka donde se leen datos desde una fuente (csv), se realiza análisis de sentimiento , se agrega la información de stream, se guarda en base de datos (mongodb) y por último se consumen datos desde una web app para mostrar tabla y gráfico de barras. A manera de facilitar el despliegue, se ha logrado crear fichero bash (.sh) para correr los subprocesos de python y abrir el navegador automáticamente con la URL del despliegue.

Deployment

Solo se tiene que ejecutar fichero bash en directorio proyecto/code
>./run_kafka_project.sh

Este fichero:
- Instala paquetes de python segun requirements.txt
- Levanta servicios a partir de docker-compose.yml
- Verifica que el servidor ksqldb esté listo para funcionar
- Corre procesos de pipeline en segundo plano
- Inicia web app con Flask
- abre el navegador con URL apropiada

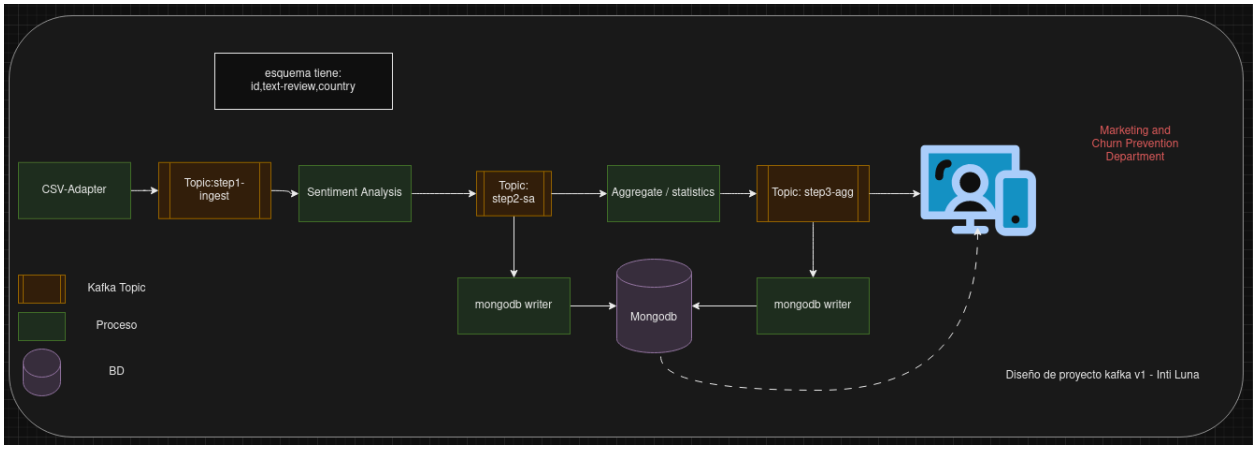
Para detener hay que:

>docker compose down
#Detener los procesos en segundo plano
>./stop_kafka_project.sh

Detalle de proyecto

Diseño

Diagramas



Servicios usados al hacer deploy

Kafka broker
Ksqldb
Zookeeper
Control Center

Consideraciones

Source-Sink
Se podría haber implementado kafka connect para leer datos de fichero csv y para escribir en base de datos (mongodb) con el plugin específico. Esta opción sería la recomendada para proyectos en producción por su fiabilidad sin embargo se requiere de más recursos. En este proyecto se decidió por un script específico dada la naturaleza del proyecto y la facilidad para ejecutarlo.

Agregación
Se ha utilizado ksqldb por su facilidad de construcción usando Ksql comparado con Kstreams usando java. Una herramienta muy útil para el desarrollo y prueba de las consultas fue la GUI de control center.

Otra manera de hacer agregaciones es hacer consultas directamente a la base de datos mongodb que contiene dos colecciones, los datos procesados de análisis de sentimiento junto a los datos originales y la colección “resumen” donde estan los datos agregados (en base a proceso de ksqldb).

Servicio a End-User
Se usó Flask porque me es familiar y porque es un framework minimalista y hay muchos plugins de extensiones.

Ficheros y descripción

Estructura

```
├── app.py
├── docker-compose.yml
├── flask.log
├── generate_syntethic_dataset.py
├── index_steps.txt
├── input_csv2.csv
├── __pycache__
│   └── app.cpython-311.pyc
├── requirements.txt
├── run_kafka_project.sh
├── static
│   ├── flujo_v1.png
│   └── styles.css
├── step1.log
├── step1.py
├── step2.log
├── step2.py
├── step3_db_writer.log
├── step3_db_writer.py
├── step3_ksqldb.log
├── step3_ksqldb.py
├── step4_save_mongodb.log
├── step4_save_mongodb.py
├── stop_kafka_project.sh
├── templates
│   └── index.html
├── tests
│   ├── step3_consumer.py
│   ├── step4_consumer_plot.py
│   ├── step4_consumer.py
│   ├── test_get_mongodb_data_full.py
│   └── test_get_mongodb_data_resumen.py
└── verificar_estado_ksqldb_server.py
```

Descripción

name	description
generate_syntethic_dataset.py	Genera fichero sintético tipo opiniones buenas, malas y neutras.
step1.py	ingesta data desde csv y crea topic "step1-ingest"
step2.py	consume datos desde topic "step1-ingest", hace análisis de sentimiento y genera topic "step2-sa"
step3_db_writer.py	consume datos de "step2-sa" y guarda en Mongodb
step3_ksqldb.py	consume datos de "step2-sa" y genera tablas agregadas y stream
step4_save_mongodb.py	consume datos de "step3-agg" y guarda en Mongodb

Adicionalmente se tiene scripts de pruebas en directorio tests. Estos scripts permiten hacer pruebas manuales y no se llegó a realizar unit tests automáticas con pytest por ejemplo.

Flujo General

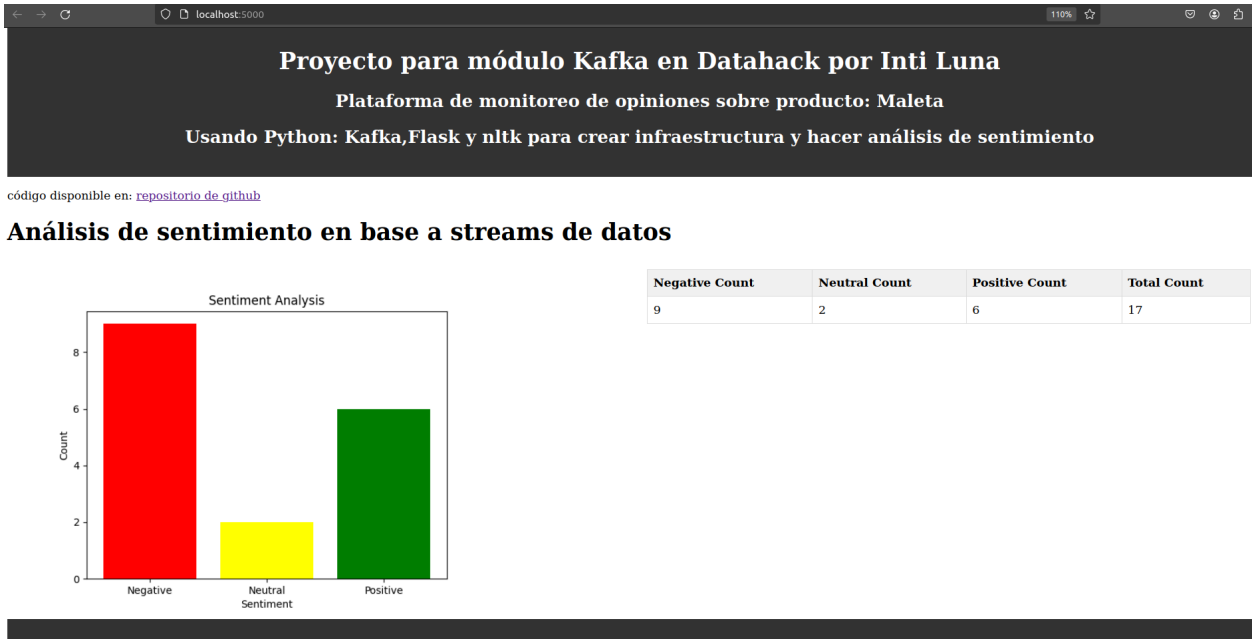
Datos de entrada

id	text	country
1	The suitcase is average in terms of design. It works as expected.	USA
2	The suitcase is average in terms of design. It works as expected.	Italy
3	I have no strong feelings about this suitcase. It is just awful.	Germany
4	Although I don't like the color, it is very versatile and I am happy with the versatile.	Spain
5	The fabrication of the product is bad. I do not not recommend it at all.	UK
6	I have no strong feelings about this suitcase. It is just bad.	Spain
7	I am not very satisfied with the suitcase. It is design but does not meet the cabin size requirements I need.	Spain
8	It is neither terrible nor construction, just an average suitcase.	Italy
9	The design is deficient and it failed after a single use. Definitely do not advise it.	Germany
10	The suitcase broke on the first use. The fabrication is very terrible and I do not discourage it at all.	UK
11	The design is horrible and it disassembled after a single use. Definitely do not propose it.	Germany
12	I am not very satisfied with the suitcase. It is design but does not meet the cabin size requirements I need.	France
13	I am very satisfied with this suitcase. It is robust, functional and meets all my expectations.	UK
14	I bought it for my son and he was delighted with. It is really versatile and solid.	Germany
15	The material of the product is awful. I do not discourage it at all.	France
16	It is a standard suitcase with material. Nothing exceptional, but it gets the job done.	Italy

Resultados

Webapp con Flask

(localhost:5000)



Análisis de sentimiento

(Usando tests/step3_consumer.py)

```
step2-sa:0:118: key=None value={
  "id": 119,
  "text": "It is very was fascinated by and firm for traveling. Without a doubt the manageable I have made.",
  "country": "France",
  "sentiment": "Positive"
}
step2-sa:0:119: key=None value={
  "id": 120,
  "text": "The design is disastrous and it ruined after a single use. Definitely do not propose it.",
  "country": "Germany",
  "sentiment": "Negative"
}
step2-sa:0:120: key=None value={
  "id": 121,
  "text": "Although I don't like the color, it is very functional and I am happy with the functional.",
  "country": "Italy",
  "sentiment": "Positive"
}
step2-sa:0:121: key=None value={
  "id": 122,
  "text": "The suitcase is very robust, although a bit small. The wheels work perfectly and it is very manageable.",
  "country": "UK",
  "sentiment": "Positive"
}
```

Oportunidades de Mejoras (no ejecutadas en v1)

- Integrar **Kafka Connect**
- Agregar datos según país.
- Encontrar otro mecanismo para generar plot de barras con python en Flask (actualmente se refresca automáticamente para generar nuevo plot).
- Implementar **Schema registry** que sería la herramienta a usar en un caso real para manejar las situaciones de cambio de esquemas en los tópicos.
- Integrar una aplicación de streaming de datos como *Reddit* o *Mastodon*
- Integrar diferentes flujos de proceso a partir de datos de topic, por ejemplo: filtrar datos según país para otros análisis y procesos (bajo la lógica que hay diferentes políticas de atención a clientes en cada región o país).
- Realizar pruebas de unit tests para mejorar la confiabilidad y mantenimiento del código.
- Agregar seguridad tanto a la web app como a cluster de kafka
- Dependiendo del caso de uso específico configurar brokers y topics para optimizar recursos, seguridad de operaciones y velocidad.

Recursos consultados

Bash
https://tecadmin.net/check-if-a-command-succeeded-in-bash/
https://unix.stackexchange.com/questions/74605/use-xdg-open-to-open-a-url-with-a-new-process
https://phoenixnap.com/kb/echo-command-linux

Kafka-Flask Integration
https://www.youtube.com/watch?v=hfi_ALPlsOQ
https://www.youtube.com/watch?v=MuLMxeI7Ytk&list=PL2UmzTIzxgL7Bq-mW--vtsM2YFF9GqhVB

Ksqldb
https://ksqldb.io/quickstart.html
https://developer.confluent.io/courses/ksqldb/intro/?_ga=2.105403331.1391371507.1717459641-1439070384.1715970528
https://docs.ksqldb.io/en/latest/developer-guide/api/
https://forum.confluent.io/t/using-ksqldb-rest-api-with-python/2732
https://github.com/bryanyang0528/ksql-python