XPodify

Ascolta la curiosità!

Lorenzo Colombo - 1081134

```python
# Leggi il dataset dei tag
tags_dataset_path = "s3://tedx-2024-colo-data/tags.csv"
tags_dataset = spark.read.option("header", "true").csv(tags_dataset_path)

# Calcola frequenza tag
tag_counts = tags_dataset.groupBy("tag").agg(count("*").alias("tag_count"))

# Ordina tag: frequenza decrescente
tag_counts = tag_counts.orderBy(col("tag_count").desc())

# Risultati
tag_counts.show()


# Esegui una join tra i tag più frequenti e il dataset originale dei tag per ottenere tutte le info
top_tags_info = tag_counts.join(tags_dataset, "tag", "left")

# Mostra i risultati con tutte le informazioni
top_tags_info.show()
#---------------
# Converte il DataFrame Spark in un DynamicFrame
tag_counts_dynamic_frame = DynamicFrame.fromDF(tag_counts, glueContext, "tag_counts_dynamic_frame")

# Opzioni per la scrittura nel database MongoDB
write_mongo_options = {
    "connectionName": "TEDX",
    "database": "unibg_tedx_2024",
    "collection": "tedx_tag_counts",
    "ssl": "true",
    "ssl.domain_match": "false"}

# Scrivi il DynamicFrame nel database MongoDB
glueContext.write_dynamic_frame.from_options(
    tag_counts_dynamic_frame,
    connection_type="mongodb",
    connection_options=write_mongo_options
```

Job 1 - PySpark

Questo job PySpark analizza i tag dei video TEDx, identificando quelli più frequenti, ordinandoli e integrando queste informazioni con il dataset originale. I risultati vengono poi salvati in un database MongoDB.

Job 2 - PySpark

Questo codice PySpark unisce i dati dei tag con i video TEDx per creare una visione aggregata, associando ogni tag a una lista di video e calcolando la frequenza di ciascun tag. I risultati vengono ordinati per frequenza e salvati in un database MongoDB per facilitarne l'accesso e l'analisi.

```python
# Leggi il dataset dei tag
tags_dataset_path = "s3://tedx-2024-colo-data/tags.csv"
tags_dataset = spark.read.option("header", "true").csv(tags_dataset_path)

# Leggi il dataset principale dei video
tedx_dataset_path = "s3://tedx-2024-colo-data/final_list.csv"
tedx_dataset = spark.read.option("header", "true").csv(tedx_dataset_path)

# Esegui una join tra i tag e i video per ottenere gli ID e i titoli dei video per ciascun tag
tag_video_info = tags_dataset.join(tedx_dataset, tags_dataset.id == tedx_dataset.id, "left") \
    .select(tags_dataset["tag"], tedx_dataset["id"].alias("video_id"), tedx_dataset["title"].alias("video_title"))

# Raggruppa per tag e crea una struttura dati all'interno di ciascun tag
tag_info = tag_video_info.groupBy("tag") \
    .agg(count("*").alias("tag_count"), collect_list(struct(col("video_id"), col("video_title"))).alias("videos"))

# Mostra i risultati
tag_info = tag_info.orderBy(col("tag_count").desc())

tag_info.show(truncate=False)
#---------------
#DataFrame Spark-->DynamicFrame
tag_counts_dynamic_frame = DynamicFrame.fromDF(tag_info, glueContext, "tag_counts_dynamic_frame")

# Opzioni per la scrittura nel database MongoDB
write_mongo_options = {
    "connectionName": "TEDX",
    "database": "unibg_tedx_2024",
    "collection": "tedx_tag_countsFromVideo",
    "ssl": "true",
    "ssl.domain_match": "false"
}

# Scrivi il DynamicFrame nel database MongoDB
glueContext.write_dynamic_frame.from_options(
    tag_counts_dynamic_frame,
    connection_type="mongodb",
    connection_options=write_mongo_options
)
```

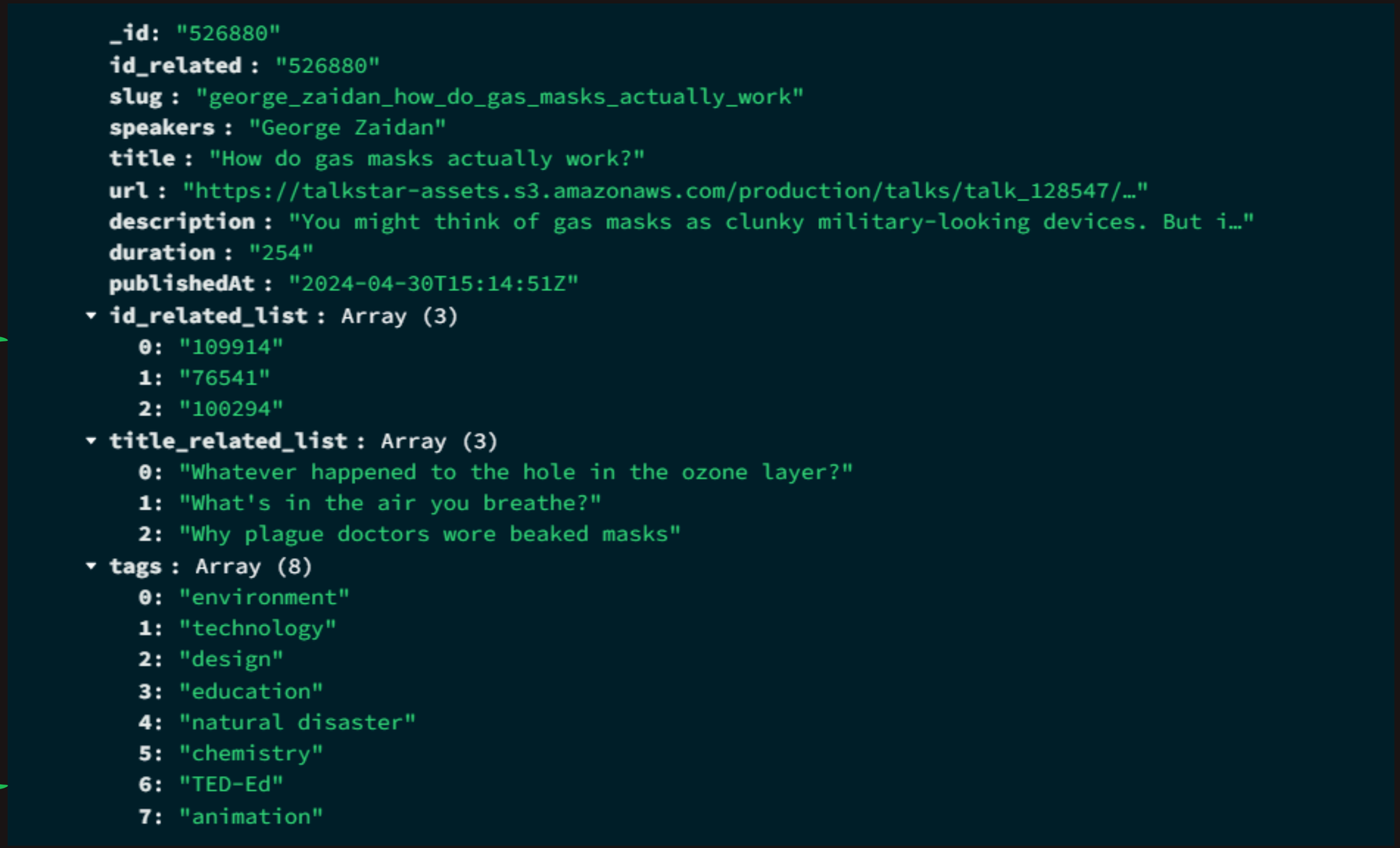
```
## READ THE RELATED VIDEOS (VERSIONE NUOVA)
related_videos_path = "s3://tedx-2024-colo-data/related_videos.csv"
related_dataset = spark.read \
    .option("header","true") \
    .option("quote", "\"") \
    .option("escape", "\"") \
    .csv(related_videos_path)

related_dataset = related_dataset.groupBy(col("id").alias("id_from")).agg(collect_list("related_id").alias("id_related_

#VERSIONE NUOVA DEL JOIN CON IL COLLEGAMENTO AI RELATIVI VIDEO
tedx_dataset_main = tedx_dataset_main.join(related_dataset, tedx_dataset_main.id == related_dataset.id_from, "left") \
    .drop("id_from") \
    .select(col("id").alias("id_related"), col("*"))
related_dataset.printSchema()

#FINE PARTE AGGIUNTA
tedx_dataset_main.printSchema()

## READ TAGS DATASET
tags_dataset_path = "s3://tedx-2024-colo-data/tags.csv"
tags_dataset = spark.read.option("header","true").csv(tags_dataset_path)


# CREATE THE AGGREGATE MODEL, ADD TAGS TO TEDX_DATASET ( Prendo tutti i tag specificati)
tags_dataset_agg = tags_dataset.groupBy(col("id").alias("id_ref")).agg(collect_list("tag").alias("tags"))
tags_dataset_agg.printSchema()
tedx_dataset_agg = tedx_dataset_main.join(tags_dataset_agg, tedx_dataset.id == tags_dataset_agg.id_ref, "left") \
    .drop("id_ref") \
    .select(col("id").alias("_id"), col("*")) \
    .drop("id") \

tedx_dataset_agg.printSchema()


write_mongo_options = {
    "connectionName": "TEDX",
    "database": "unibg_tedx_2024",
    "collection": "tedx_data",
    "ssl": "true",
    "ssl.domain_match": "false"}
from awsglue.dynamicframe import DynamicFrame
tedx_dataset_dynamic_frame = DynamicFrame.fromDF(tedx_dataset_agg, glueContext, "nested")
```

WatchNext - PySpark

Questo processo raccoglie e unisce informazioni sui TED Talks, sfruttando i dati dei contenuti correlati, creando un database completo e integrato.

RISULTATI:

Dati Elaborati



POSSIBILI EVOLUZIONI:

Web Scraping

Offrire tag maggiormente personalizzati in base alle parole utilizzate nei video

Job più complessi

Utilizzo di tecniche di aggregazione e filtri più avanzati

Integrazione di Grafici

Inserire diagrammi per comprendere meglio l’interazione tra i diversi tag

Riferimenti

Trello

Github