



A large black smartphone is positioned vertically. On its screen, the 'TEDx LANGUAGE' logo is displayed in red, with 'TEDx' in a larger font above 'Language'. Below the logo, three names and IDs are listed: 'Biscaro Alessandro - 1087892', 'Fabbris Thomas - 1086063', and 'Gambirasio Lorenzo Umberto - 1087441'. A cartoon illustration of a man with brown hair, wearing a blue t-shirt and brown pants, stands to the left of the phone, pointing his right index finger towards the phone's screen. The background behind the phone is a light grey gradient.

HOMEWORK 2

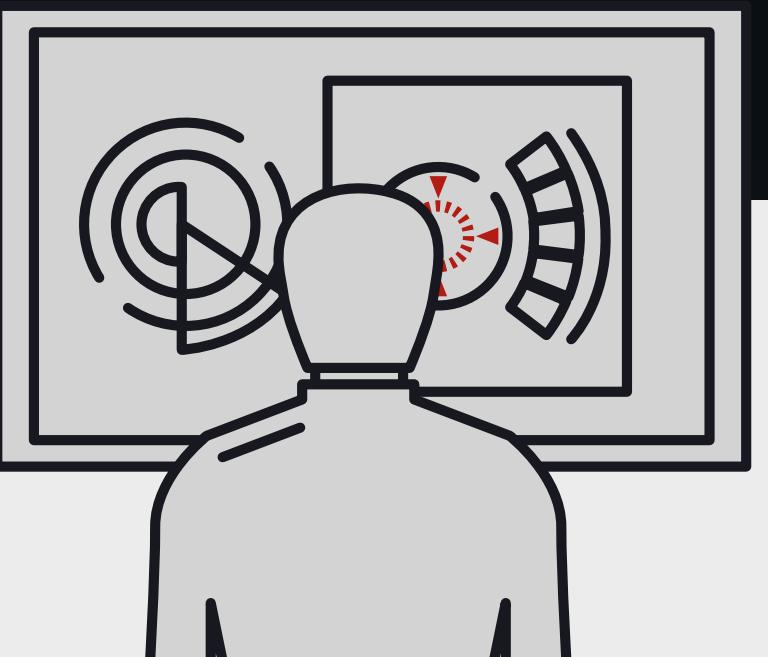
Piattaforme Cloud e
Applicazioni Mobili

A.A 2024-2025

TedXJob - WatchNext

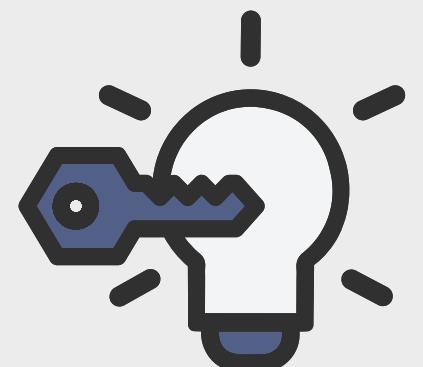


```
88 # READ RELATED VIDEOS
89 watch_next_dataset_path = "s3://tedx-2025-data-tf-20250315/related_videos.csv"
90 watch_next_dataset = spark.read \
91   .option("header", "true") \
92   .option("quote", "") \
93   .option("escape", "") \
94   .csv(watch_next_dataset_path)
95
96 internal_to_id_df = watch_next_dataset.select("internalId", "id").dropna().dropDuplicates()
97 internal_to_id_df = internal_to_id_df.withColumnRenamed("internalId", "related_id").withColumnRenamed("id", "related_video_id")
98
99 related_enriched_df = watch_next_dataset.join(
100   internal_to_id_df,
101   on="related_id",
102   how="left"
103 ).select(
104   watch_next_dataset["id"].alias("main_id"),
105   "related_video_id"
106 )
107
108 related_grouped_df = related_enriched_df.groupBy("main_id").agg(
109   collect_list("related_video_id").alias("related_video_ids")
110 )
111
112 tedx_dataset_final = tedx_dataset_agg.join(
113   related_grouped_df,
114   tedx_dataset_agg["_id"] == related_grouped_df["main_id"],
115   "left"
116 ).drop("main_id")
```



La modifica allo schema proposta è l'aggiunta al documento di un campo related_videos_id contenente un'array di id di video correlati.

Poichè non esiste un'associazione diretta tra il dataset principale e il dataset watch_next attraverso uno o più attributi, abbiamo deciso di collegare i talk sulla base di internal_id e related_id nel dataset watch_next e di ricercare il video corrispondente nel dataset originale.



OBIETTIVO



Abbiamo modificato il job Glue "TedXJob" in modo da aggiungere al dataset originale le informazioni riguardanti i video correlati.

Questa soluzione si propone come un modo naturale per implementare un sistema di personalizzazione dei contenuti molto semplice, ma efficace.

L'arricchimento dei dati a disposizione potrebbe rivelarsi utile al fine di migliorare l'esperienza utente. Si potrebbe mostrare a fianco del video in esecuzione una lista di talk che trattano della medesima tematica, in modo da consentire all'utente di approfondire la sua conoscenza in un determinato ambito mentre sta rifinendo l'apprendimento di una lingua straniera.



DATA CLEANING



Il secondo job si è concentrato sulla pulizia dei dati, in modo da identificare errori e inconsistenze nel dataset, allo scopo di eliminare i record non validi. L'output del processo di pulizia dei dati è stato una nuova collezione MongoDB denominata tedx_data_cleaned.

```
# Normalizzazione del testo, rimuovendo spazi extra, caratteri di controllo e convertendo in lowercase
def normalize_text_func(text):
    if text is None:
        return None
    text = text.lower()
    text = re.sub(r'[\r\n\t]+', ' ', text)
    text = text.strip()
    return text

normalize_text_udf = udf(normalize_text_func, StringType())
```

Normalizzazione del testo (campi description, slug, speaker), convertendolo in minuscolo e rimuovendi spazi o ritorni a capo superflui

```
# Funzione per validare URL
def is_valid_url_regex(url_string):
    if url_string is None:
        return False
    return bool(re.match(r'^^(https?|ftp):\/\/[^\s/$.?#].[^\s]*$', str(url_string)))
```

Validazione URL con regex

```
transformed = df_transformed \
    .withColumn(
        "publishedAt_formatted",
        when(col("publishedAt_ts").isNotNull(), date_format(col("publishedAt_ts"), "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"))
        .otherwise(lit("1970-01-01T00:00:00.000Z"))
    ) \
    .withColumn("url_is_valid", is_valid_url_udf(col("url")))
    .withColumn("related_videos_are_valid", validate_related_ids_udf(col("related_video_ids")))
```

Verifica dell'effettiva esistenza dei video correlati ad un talk e formattazione del campo publishedAt nel formato data ISO 8601.

```
# Filtraggio finale per ID validi, URL validi e related_videos validi
df_valid = df_cleaned.filter(
    (col("_id").isNotNull()) &
    (col("duration").isNotNull()) &
    (col("url_is_valid") == True) &
    (col("related_videos_are_valid") == True)
)
```

Rimozione dei talk con id nullo oppure duplicato

Si è passati da 7055 documenti componenti la collezione MongoDB a 6092 documenti a seguito delle operazioni di data cleaning e data presentation, con una sostanziale assenza di valori nulli nel dataset

SCHEMA DEL DATABASE



tedx_data		
_id	string	NN
title	string	
slug	string	
description	string	
duration	int	
publishedAt	date	
speakers	string	
url	string	
related_video_ids	string[]	
tags	string[]	

Qui è illustrata la struttura della collezione principale tedx_data_cleaned, utilizzata per archiviare le informazioni di ciascun TEDx Talk.

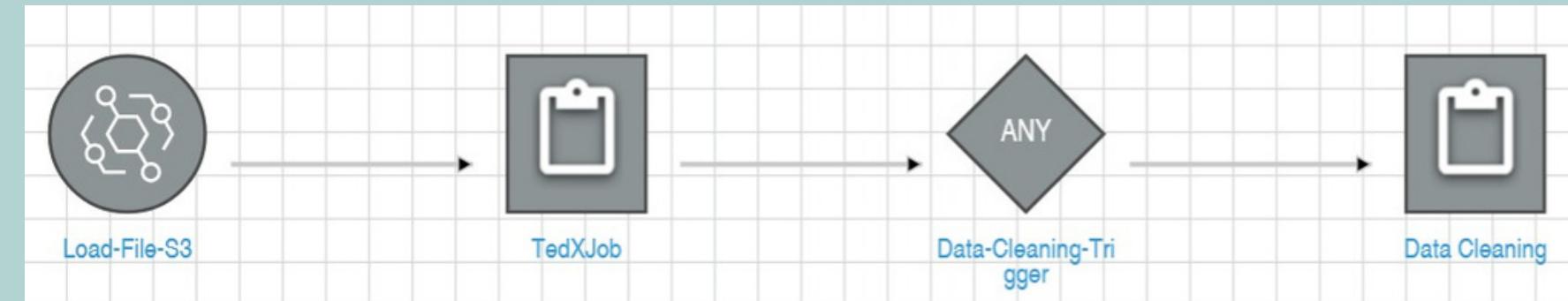
INTEGRAZIONE TRA JOB GLUE



Il principale inconveniente derivante dalla presenza di due job separati era la necessità di dover lanciare manualmente la loro esecuzione al caricamento di nuovi file su un bucket S3 dedicato.

Per superare tale limitazione, è stato definito un workflow in AWS Glue al fine di definire una ETL pipeline per il progetto, composta dai due job presentati in precedenza. In particolare, il workflow viene attivato da una regola EventBridge e scatena l'esecuzione di TedXJob e Data Cleaning.

```
[  
  "source": ["aws.s3"],  
  "detail-type": ["Object Created"],  
  "detail": {  
    "bucket": {  
      "name": ["tedx-2025-data-tf-20250315"]  
    },  
    "object": {  
      "key": [  
        {"prefix": "final_list.csv"},  
        {"prefix": "details.csv"},  
        {"prefix": "tags.csv"},  
        {"prefix": "related_videos.csv"}  
      ]  
    }  
  }  
]
```



Pattern della regola EventBridge

CRITICITÀ TECNICHE

LOGICA DELLO SCHEMA

Individuazione della logica per collegare il dataset preesistente e il dataset watch_next

GESTIONE CAMPI NULLI

Difficoltà nell'eliminazione dei record con durata nulla

SOVRACCARICO API

Elevato numero di chiamate API per recuperare i dettagli e le informazioni dei video correlati



POSSIBILI EVOLUZIONI



Espansione dello schema mediante tecniche di web scraping (es. recuperare informazioni aggiuntive sui talk come la località in cui sono stati tenuti)



Arricchimento del dataset mediante tecniche di web scraping (es. recuperare le trascrizioni dei talk fornite da TedX)



Aggiunta di un campo di tipo strutturato per related_videos, in modo da aver a disposizione sin da subito tutti i dati senza dover ricorrere a chiamate API addizionali

LINK utili



[GitHub](#)



[Trello](#)

