

UNITEDx

Trova il percorso di studi più adatto
ai tuoi interessi.



Marco Abbadini (1048650)
Lorenzo Conti (1046163)
Fabio Sangregorio (1046566)



Job Pyspark base

Job AWS Glue: AddWatchNext

```
1 ##### READ INPUT FILES TO CREATE AN INPUT DATASET
2 watch_next_dataset = spark.read(...)
3
4 ##### DROP URL COLUMN
5 watch_next_dataset = watch_next_dataset.drop("url")
6
7 ##### AGGREGATE ITEMS UNDER SAME INDEX
8 watch_next_dataset_agg = watch_next_dataset
9     .groupBy(col("idx").alias("idx_ref"))
10     .agg(collect_set("watch_next_idx").alias("watch_next"))
11
12 ### READ TEDx DATASET FROM MONGODB
13 tedx_dataset = glueContext.create_dynamic_frame.from_options(
14     connection_type="mongodb",
15     connection_options=read_mongo_options).toDF()
16
17 # CREATE THE AGGREGATE MODEL, ADD WATCHNEXT TO TEDx_DATASET
18 tedx_dataset_agg = tedx_dataset.join(
19     watch_next_dataset_agg, tedx_dataset.id == watch_next_dataset_agg.idx_ref, "left"
20 ).drop("idx_ref")
21
22 # WRITE RESULT TO MONGODB
23 tedx_dataset_dynamic_frame = DynamicFrame.fromDF(tedx_dataset_agg, glueContext, "nested")
```

Codice completo su Github

Job AWS Glue: CreateTagsCollection

```
1 tags_dataset_agg = tags_dataset.groupBy(col("tag"))
2     .agg(collect_set("idx")
3     .alias("talk_id"))
```

Job CreateDataLake:

- Utilizzo di `.option("multiline", "true")` per gestire le descrizioni su più righe nel CSV

Job CreateTagsCollection:

- Uso di `collect_set` per evitare i duplicati

Job AddWatchNext:

1. Estrae i dati da MongoDB
2. Estrae i "watch next" dal relativo CSV
3. Aggiunge il campo `watch_next` ai dati di MongoDB
4. Aggiorna il risultato su MongoDB

MongoDB

Esempio di elemento nella collection **tedx_data**

```
_id: "8d2005ec35280deb6a438dc87b225f89"
details: "More barriers exist now than at the end of World War II, says designer..."
main_speaker: "Alexandra Auer"
posted: "Posted Apr 2020"
tags: Array
  0: "TED"
  1: "talks"
  2: "design"
  3: "society"
  4: "identity"
  5: "social change"
  6: "community"
  7: "humanity"
  8: "TEDx"
title: "The intangible effects of walls"
url: "https://www.ted.com/talks/alexandra_auer_the_intangible_effects_of_wal..."
watch_next: Array
  0: "5bd34fcc55d9e1267f605fa0c060d54e"
  1: "8576654442b6633b1dc0eb48a989172a"
  2: "078766d6cc461cf71d45dc268b66db95"
  3: "d9896b41b372ec60cdd3c662e57caad3"
  4: "9f7b1654e792011b7e1c6f4288520226"
  5: "fe35edd737282ab3a325f2387cf1b50b"
  6: "5134ae81a27c94354173f38e84289ad5"
```

Il job **CreateTagsCollection** aggiunge, per ogni tag, i video correlati. Questi possono essere utilizzati per consigliare i talk più inerenti con un corso in base, per esempio, ai tag in comune.

Esempio di elemento nella collection **tedx_tags**

```
_id: ObjectId("60a51e49d24768101bee5fa3")
tag: "blockchain"
talk_id: Array
  0: "21d128218d4b339848dc096d0e062cc4"
  1: "ee5d897eb842b1f92ffe3c30207eef78"
  2: "39970e32cc3df972c9a2d910d800e89b"
  3: "4d8f2cde82fb05343a6f72c42b7965c4"
  4: "abe2b18b8ef5c399f1e4ac5d43133288"
  5: "09f05b22600f0481da31e52558c50b11"
  6: "9ea465c26bbd33fb352b82a01b4f38a1"
  7: "2a695343b46e1547b8675d215c8ea056"
  8: "3a00b0d011a79ab1838728944c199e57"
```

Criticità

Debugging

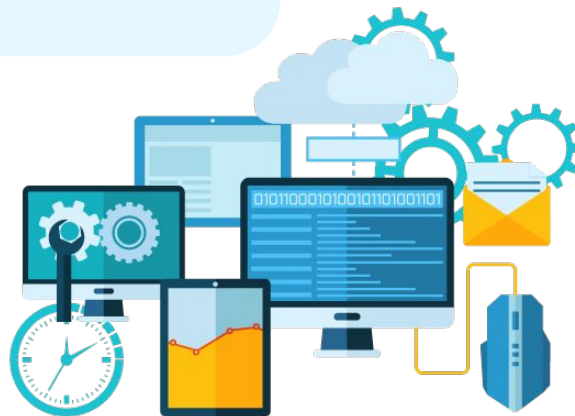
Non è stato trovato uno strumento/modo semplice per debuggare il codice da errori. I log disponibili sono risultati poco efficaci e “confusionari”.

Dati duplicati

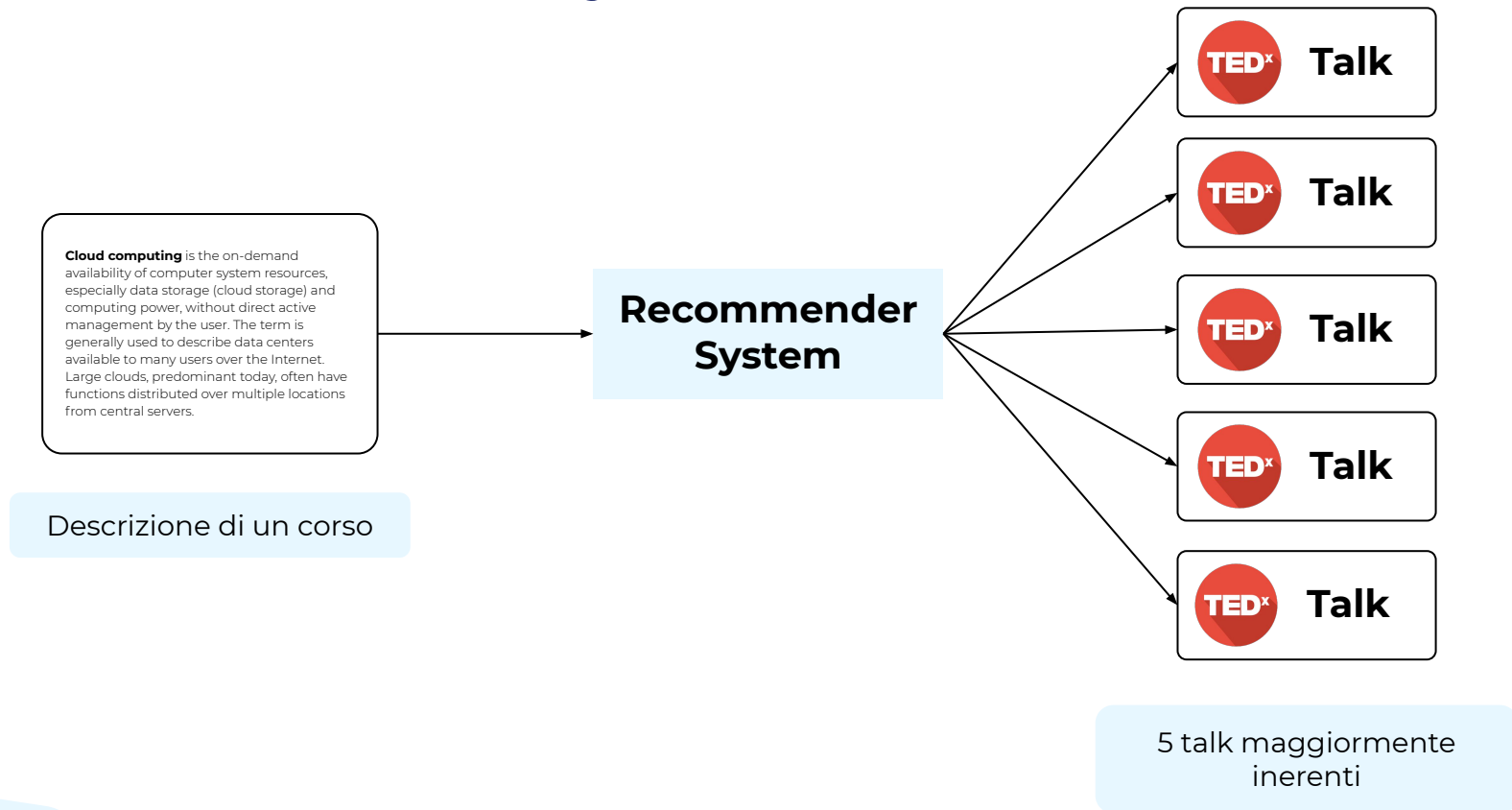
I dati di alcuni video consigliati come watch next sono duplicati. Il problema è stato risolto utilizzando un `collect_set`.

Formato dati

Alcuni dati non presentano il corretto numero di views o sono mancanti di speaker. Alcuni record presentano descrizioni su più righe (inframezzate da `\n`), che necessitano di una gestione apposita (opzione “multiline” nella read)



Recommender System



Recommender System: Nearest Neighbour

Job AWS Glue: CreateNNModel

```
df = tedx_dataset.select("*").toPandas()

count_vectorizer = TfidfVectorizer(
    analyzer="word", preprocessor=None, stop_words='english', max_features=None)

tfidf = count_vectorizer.fit(df['details'])
tfidf_matrix = count_vectorizer.transform(df['details'])

nn = NearestNeighbors(n_neighbors=5, n_jobs=-1)
nn.fit(tfidf_matrix)

# Save recommender system
s3.Bucket('<Bucket Path>').put_object(Key="model.pkl", Body=fp.read())

# Save vectorizer
s3.Bucket('<Bucket Path>').put_object(Key="vectorizer.pkl", Body=fp.read())




# Save id_map
idxs = list(df["idx"].values)
mapped_list = { i: idxs[i] for i in range(len(idxs))}
s3.Bucket('<Bucket Path>').put_object(Key="id_map.json", Body=fp.read())
```

Si ottiene una versione vettorizzata delle descrizioni dei talk

Si crea un modello Nearest Neighbour che, ricevendo in input un testo descrittivo, presenta i 5 talk maggiormente inerenti

Si salvano vettorizzatore e modello nel bucket S3

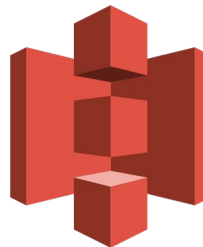
S3 Bucket: tcm-model-1

Name	Type	Last modified	Size
 id_map.json	json	May 29, 2021, 16:55:47 (UTC+02:00)	195.3 KB
 model.pkl	pkl	May 29, 2021, 16:55:47 (UTC+02:00)	1.8 MB
 vectorizer.pkl	pkl	May 29, 2021, 16:55:47 (UTC+02:00)	1.8 MB

Vantaggi e criticità

Il modello è costruito a partire dalla descrizione di tutti i talk. Con l'ottica di utilizzare il modello tramite invocazioni di lambda function, risulterebbe oneroso caricare l'intero dataset ed addestrare un modello ad ogni invocazione.

Vantaggi



Criticità

È necessario creare e salvare un nuovo modello ed un nuovo vettorizzatore ogni volta che nuovi talk sono inseriti nel database. Si può pensare di effettuare inserimenti batch per rendere il processo meno oneroso.

UNITEDx



[Board](#)

[GitHub](#)

