TEDx-Perience

Pyspark & AWS Glue

Cremonesi Andrea (1074260) - Nigro Marco (1046992)

Lettura “watch_next_dataset”

Abbiamo ordinato le righe di **watch_next_dataset** in base a **num_views** in ordine decrescente.

Abbiamo raggruppato i dati, ora ordinati, di **watch_next_dataset** in base all’id del video corrispondente.

```
watch_next_dataset_path = "s3://unibg-tedx-data-2023-test-mnigro/watch_next_dataset.csv"
watch_next_dataset = spark.read.option("header", "true").csv(watch_next_dataset_path)

# Unione 'watch_next_dataset' con 'tedx_dataset' per ottenere 'num_views'
watch_next_dataset = watch_next_dataset.join(tedx_dataset, watch_next_dataset.watch_next_idx == tedx_dataset.idx, "inner") \
    .select(watch_next_dataset.idx, watch_next_dataset.watch_next_idx, tedx_dataset.num_views)

# Ordina 'watch_next_dataset' per 'num_views' e aggrega per 'idx'
watch_next_dataset_agg = (
    watch_next_dataset.groupBy(col("idx").alias("idx_ref"))
    .agg(
        sort_array(collect_set(struct("num_views", "watch_next_idx")), asc=False)
        .alias("collected_list")
    )
    .withColumn("watch_next_list", col("collected_list.watch_next_idx"))
    .drop("collected_list")
)
```

Calcolo punteggio medio dei video

Abbiamo calcolato il punteggio medio (**avg_points**) per ogni video come media ponderata tra il numero di visualizzazioni e il numero di like normalizzati, estratti dal dataset **data.csv**.

Per effettuare questo calcolo abbiamo definito la funzione **calc_avg_points**

```
## LEGGI IL FILE DATA E CALCOLA IL PUNTEGGIO MEDIO DEI VIDEO

tedx_data_path = "s3://unibg-tedx-data-2023-test-mnigro/data.csv"
tedx_data = spark.read \
    .option("header", "true") \
    .option("quote", "\"") \
    .option("escape", "\"") \
    .option("delimiter", ",") \
    .csv(tedx_data_path)

tedx_data = tedx_data.join(tedx_dataset, tedx_data.title == tedx_dataset.title, "inner") \
    .select(tedx_dataset.idx, tedx_dataset.num_views, tedx_data.likes)

# Salva i dati delle colonne likes e views come liste
likes = tedx_data.select("likes").rdd.flatMap(lambda x: x).map(lambda x: int(x) if x else 0).collect()
views =  tedx_data.select("num_views").rdd.flatMap(lambda x: x).map(lambda x: int(x) if x else 0).collect()

# Normalizzazione dei valori delle liste
n_likes = [x/max(likes) for x in likes]
n_views = [x/max(views) for x in views]

# Calcolo della lista dei valori
avg_points_list = calc_avg_points([(n_likes, 80), (n_views, 20)])

# Creazione Data Frame con valori calcolati
df_avg_points = spark \
    .createDataFrame([(avg_points_list,)], ['lista'])\
    .select(explode(col('lista')).alias('avg_points'))

# Aggiunta degli indici ai Data Frame per eseguire la join
df_avg_points = df_avg_points.withColumn("index", row_number().over(Window.orderBy(monotonically_increasing_id())))
tedx_data = tedx_data.withColumn("index", row_number().over(Window.orderBy(monotonically_increasing_id())))

# Unione dei Data Frame
tedx_data_complete = tedx_data \
    .join(df_avg_points, tedx_data.index == df_avg_points.index, "inner") \
    .drop("index") \
    .withColumnRenamed("idx", "idx_ref") \
```

Funzione Media Ponderata

```python
def calc_avg_points(data):
    # Crea una lista contenente le liste di dati
    lists = [x[0] for x in data]
    # Crea una lista contenente i pesi relativi alle liste di dati
    weights = [x[1] for x in data]
    # Calcola la somma dei pesi
    tot_weights = sum(x for x in weights)

    # Trova la lunghezza minima tra tutte le liste di dati
    min_length = min([len(x) for x in lists])

    # Crea una lista vuota per contenere i valori medi
    avg_points_list = []
    # Calcola la media per ogni elemento
    for i in range(0, min_length):
        mean = 0
        # Calcola la media di ogni elemento usando il peso corrispondente nella lista weights
        for j in range(0, len(weights)):
            mean = mean + lists[j][i] * weights[j]
        # Aggiungi la media pesata alla lista di valori medi
        avg_points_list.append(mean / tot_weights * 100)

    # Restituisci la lista di medie
    return avg_points_list
```

Si occupa di assegnare il punteggio a ogni video calcolando la media ponderata tra numero di like e visualizzazioni

- **data**: lista di tuple contenenti la lista dei valori delle colonne interessate e il loro rispettivo peso.
- **avg_points_list**: lista dei punteggi medi relativi ad ogni video.

Aggiorna Dataset

Abbiamo aggiunto a **tedx_dataset** i dati relativi a:

- **punteggio medio**
- **tags**
- **numero di like**
- **video consigliati** (ordinati per numero di visualizzazioni)

```
## AGGIORNA DATASET

tedx_dataset_agg = tedx_dataset \
    .join(tedx_data_complete, tedx_dataset.idx == tedx_data_complete.idx_ref, "inner") \
    .join(tags_dataset_agg, tedx_dataset.idx == tags_dataset_agg.idx_ref, "left") \
    .join(watch_next_dataset_agg, tedx_dataset.idx == watch_next_dataset_agg.idx_ref, "left") \
    .drop("idx_ref") \
    .select(col("idx").alias("_id"), col("*")) \
    .drop("idx") \
```

Visualizzazione dei dati in MongoDB

- **tags**: lista dei tag del video
- **watch_next_list**: id dei video consigliati ordinati per numero di visualizzazione
- **likes**: numero di like del video
- **avg_points**: punteggio del video

Criticità tecniche

- **Formattazione errata dei valori di num_views in tedx_dataset:**
  Abbiamo eliminato le virgole presenti nei valori del campo **num_views** e ri-formattato i valori, convertendoli in numeri interi.
- **Dati duplicati presenti in watch_next_dataset:**
  Abbiamo utilizzato il comando **collect_set** per eliminare i duplicati presenti nel dataset.
- **Lettura dei dataset:**
  La differenza di formato dei dataset ha implicato la definizione di opzioni di lettura specifiche per i diversi per ognuno di essi.
- **Dataset obsoleti**:
  I dataset non vengono periodicamente aggiornati, perciò i dati diventano obsoleti nel tempo.
- **Mancanza di dati**:
  In alcuni video il numero di visualizzazioni è pari a 0. Questo porta ad un calcolo errato del valore **avg_points**.

Possibili Evoluzioni

- Implementazione di uno **scraper** per mantenere i dati costantemente aggiornati
- Ottimizzazione della funzione che calcola il punteggio attribuito ai video

Link utili