Homework 2 - Big Data Engineering

Anna Lamboglia M63/001219

14 maggio 2022

Indice

1 Traccia

The second homework concerns the development of a program based on Apache PIG for extracting some useful information from the review file of YELP dataset (https://www.yelp.com/dataset). The output of the analysis must be reported within a pdf file of 8/10 pages (whose name will be namesurnamehomework2), whose structure should include the following information:

- Problem introduction and dataset characterization;
- Processing queries and algorithms based on Apache Pig with the related discussion

This homework can be done individually or in groups of up to two people. In the case of groups, the output must be at least 10 pages. The deadline for this homework is Monday May 16, 2022.

2 Introduzione al problema e caratterizzazione del Dataset

L'obiettivo di questo elaborato consiste nell'analizzare le recensioni della piattaforma Yelp. Yelp è un social network ed una guida online dove persone, prevalentemente locali, forniscono consigli utili e si scambiano opinioni riguardo a posti ed attività caratteristiche dei luoghi verso cui viaggiano, per lavoro o per vacanza.

Il dataset è stato fornito dalla Yelp a questo link: `https://www.yelp.com/dataset`. Esso è formato da 5 file JSON, ma per analizzare le recensioni si farà uso solo dei seguenti due file:

- *Business.json:* contiene le informazioni dei business registrati a Yelp. I dati includono business id, nome, indirizzo, valutazioni, numero di recensioni, attributi (come ad esempio accesso per sedie a rotelle, cani ammessi, Wi-fi, ecc) che possono variare, categorie, orari di apertura, numero di recensioni ricevute, posizione e così via come mostrato nel codice[1]. La grandezza del file è di 113 Mb;
- *Review.json:* contiene tutte le informazioni relative alle recensioni di un dato business. Sono presenti i campi id recensione, id utente (ossia l'utente che ha scritto la recensione), business id (ossia il business che ha ricevuto la recensione), stelle, data della recensione, contenuto del testo e alcune altre caratteristiche come useful, cool e funny [2]. La grandezza del file è di 4.97 Gb;

```
{'business_id': 'Pns2l4eNsfO8kk83dixA6A',
 'name': 'Abby Rappoport, LAC, CMQ',
 'address': '1616 Chapala St, Ste 2',
 'city': 'Santa Barbara',
 'state': 'CA',
 'postal_code': '93101',
 'latitude': 34.4266787,
 'longitude': -119.7111968,
 'stars': 5.0,
 'review_count': 7,
 'is_open': 0,
 'attributes': {'ByAppointmentOnly': 'True'},
```

```
 'categories': 'Doctors, Traditional Chinese Medicine, Naturopathic/Holistic,
    ↪ Acupuncture, Health & Medical, Nutritionists',
 'hours': None}
```

Listing 1: Esempio Business.json

```
{'review_id': 'KU_O5udG6zpxOg-VcAEodg',
 'user_id': 'mh_-eMZ6K5RLWhZyISBhwA',
 'business_id': 'XQfwVwDr-v0ZS3_CbbE5Xw',
 'stars': 3.0,
 'useful': 0,
 'funny': 0,
 'cool': 0,
 'text': "If you decide to eat here, just be aware it is going to take about 2
    ↪ hours from beginning to end. We have tried it multiple times, because I
    ↪ want to like it! I have been to it's other locations in NJ and never had a
    ↪  bad experience. \n\nThe food is good, but it takes a very long time to
    ↪ come out. The waitstaff is very young, but usually pleasant. We have just
    ↪ had too many experiences where we spent way too long waiting. We usually
    ↪ opt for another diner or restaurant on the weekends, in order to be done
    ↪ quicker.",
 'date': '2018-07-07 22:09:11'}
```

Listing 2: Esempio Review.json

3 Rappresentazione del modello dei dati

Come richiesto dalle specifiche di progetto, è stata utilizzata Apache Pig, ossia una piattaforma che consente di analizzare grandi insiemi di dati. Pig permette di adoperare un linguaggio di alto livello, chiamato Pig Latin che, abbinato ad un'infrastruttura che consente implementazioni parallele come Hadoop, rende possibile l'analisi di dati di grandi dimensioni.

Le caratteristiche principali di Pig sono le seguenti:

- **Facilità di programmazione.** Pig Latin è un linguaggio molto simile all'SQL, quindi permette anche a chi è familiare con il linguaggio SQL di poter lavorare senza troppe problematiche;
- **Operazioni Built-in.** Pig è dotato di una serie di funzioni integrate, come le funzioni *eval*, *load/store*,*math*, *string*, *bag* e *tuple*;
- **Flessibilità.** Apache Pig analizza tutti i tipi di dati, sia strutturati che non strutturati;
- **Ottimizzazione.** Il modo in cui i jobs sono codificati permette al sistema di ottimizzare automaticamente la loro esecuzione, consentendo all'utente di concentrarsi sulla semantica piuttosto che sull'efficienza;
- **Estensibilità.** Gli utenti possono creare le proprie funzioni, ossia le User defined functions (UDFs), per eseguire elaborazioni speciali. Le UDF Pig possono attualmente essere implementate in sei linguaggi: Java, Jython, Python, JavaScript, Ruby e Groovy.

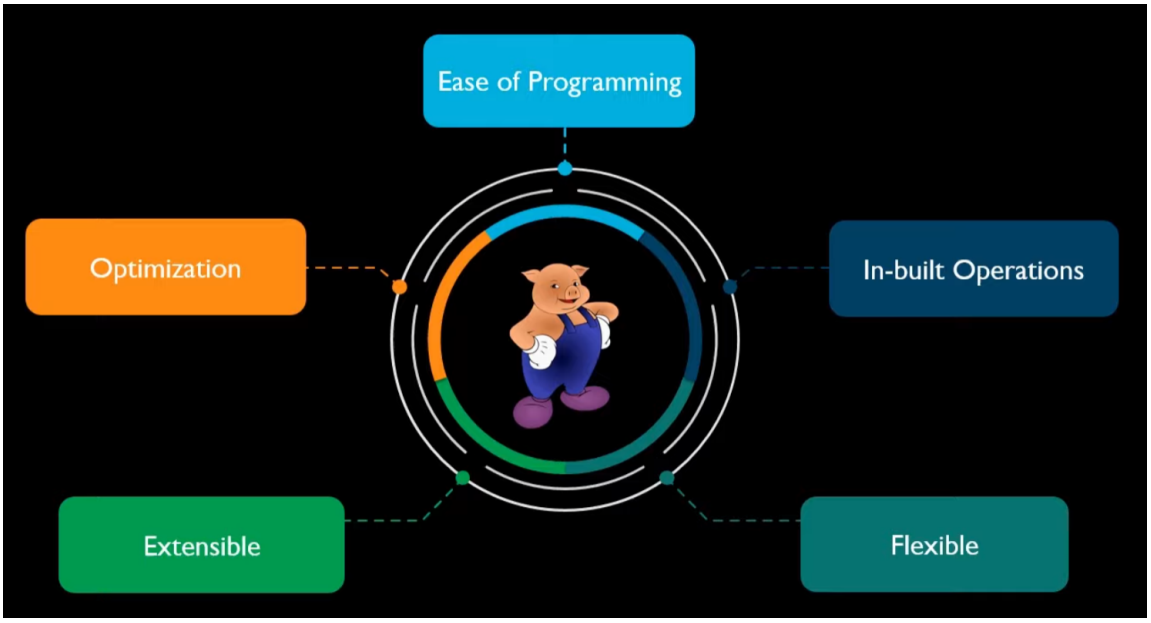


Figura 1: Caratteristiche di Pig

Come già citato, Pig viene spesso utilizzato in combinazione con Hadoop, che consente in un framework concepito per scrivere facilmente applicazioni che elaborano grandi quantità di dati in parallelo su cluster di grandi dimensioni, assicurando un'elevata affidabilità e disponibilità. I vantaggi nell'utilizzo di Hadoop dipendono dalle sue caratteristiche principali ossia:

- La possibilità di avere un file system distribuito chiamato **HDFS**, progettato per immagazzinare una enorme quantità di dati, in modo da ottimizzare le operazioni di accesso ed archiviazione ad un numero ristretto di file di grandi dimensioni, differenziandosi dai file system tradizionali ottimizzati per gestire grandi quantità di file di piccole dimensioni;

- Il modello di programmazione **MapReduce**, ossia un framework di esecuzione per l'elaborazione parallela di applicazioni batch. Si basa sul principio in cui invece di trasferire i dati al sistema computazionale, HDFS elabora direttamente i data nodes e la trasformazione di tali dati viene realizzata in loco, attraverso MapReduce. Ogni nodo elabora i dati in base alla richiesta e successivamente inoltra i risultati. Questi ultimi vengono consolidati su un nodo master, il quale si occupa anche di memorizzare tutti i metadati associati alla gestione dei cluster.

3.1 Preprocessing

Dato che il file *Review.json* risulta molto pesante(circa 4 Gb), si è pensato di limitare il numero di recensioni; delle 6,990,280 totali, ne sono state analizzate 10,000.

Dunque, dopo aver caricato il file JSON tramite la funzione built in di LOAD, è stato utilizzato il comando LIMIT per limitare il numero di recensioni prese in fase di analisi.

```
Review_cut = LIMIT R 10000;
```

4 Query

Pig permette di lavorare in due modalità:

- *Local Mode:* tutte le operazioni vengono eseguite in una singola JVM ed i dati di input ed output si trovano localmente;
- *Hadoop Mode:* è possibile utilizzare la divisione in cluster grazie ad Hadoop ed i file di input ed output si trovano su nodi HDFS.

Per l'esecuzione delle query all'interno di questo elaborato, si è lavorato in Local Mode e, per definirlo, è stato necessario il seguente comando sulla shell Linux:

```
./pig -x local
```

Listing 3: Local Mode

Le query sono state elaborate ipotizzando di trovarsi nella prospettiva della piattaforma Yelp, effettuando delle valutazioni sul totale delle recensioni invece che concentrarsi su un singolo business.

4.1 Query 1

La prima query consiste nell' *ordinare le parole presenti nelle recensioni che non siano stopwords in ordine decrescente*. Una elaborazione simile può essere utile per comprendere quali sono le parole più utilizzate nelle recensioni degli utenti, per valutare, nel caso si voglia realizzare sentiment analysis, quali possono essere le parole interessanti da analizzare.

```
R = LOAD '/media/sf_Challenge/yelp_academic_dataset_review.json' USING JsonLoader
    ↪ ('review_id: chararray, user_id: chararray, business_id: chararray, stars:
    ↪ float, useful: chararray, funny: chararray, cool: chararray, text:
    ↪ chararray, date: chararray');

Review_cut = LIMIT R 10000;
stoplist = LOAD '/media/sf_Challenge/stopwords.txt' USING TextLoader AS (stop:
    ↪ CHARARRAY);
words = FOREACH Review_cut GENERATE FLATTEN(TOKENIZE(REPLACE(LOWER(TRIM(text))
    ↪ ,'[\\p{Punct},\\p{Cntrl}]',''))) AS word;
words = JOIN words BY word LEFT, stoplist BY stop;
words = FILTER words BY stoplist::stop IS NULL;
grpd = GROUP words BY $0;
cntd = FOREACH grpd GENERATE $0, COUNT($1);
unmix = ORDER cntd BY $1 DESC, $0 ASC;
STORE unmix INTO '/media/sf_Challenge/outputword' USING PigStorage (',');
```

Listing 4: Codice Prima Query

```
food,4984
time,3135
service,3106
nice,1791
dont,1701
im,1624
staff,1539
ive,1517
didnt,1481
delicious,1429
try,1412
restaurant,1397
love,1386
chicken,1379
friendly,1363
definitely,1350
little,1349
experience,1257
people,1256
menu,1250
amazing,1214
wait,1079
day,1060
recommend,1052
bar,1033
cheese,994
pretty,980
pizza,958
am,946
fresh,940
eat,904
night,898
told,898
minutes,869
sauce,849
```

Figura 2: Risultato Query 1

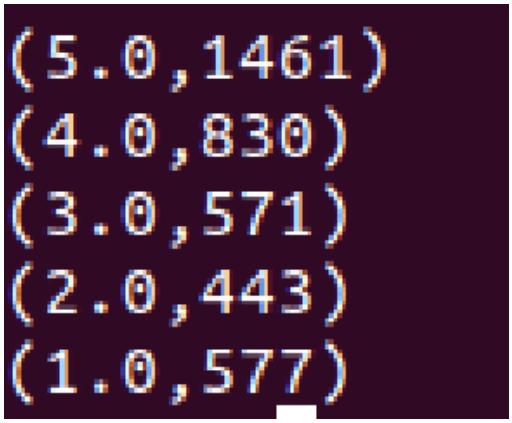
4.2 Query 2

Per l'ideazione della seguente query è stato sfruttato il risultato della Query 1.In particolare, effettuando un'analisi sulle due parole più frequenti, si vuole associare, ad una specifica parola, il conteggio di recensioni per ogni valutazione, con l'obiettivo di etichettare quella determinata parola ad un sentiment che può essere *positivo, negativo* o *neutro*.

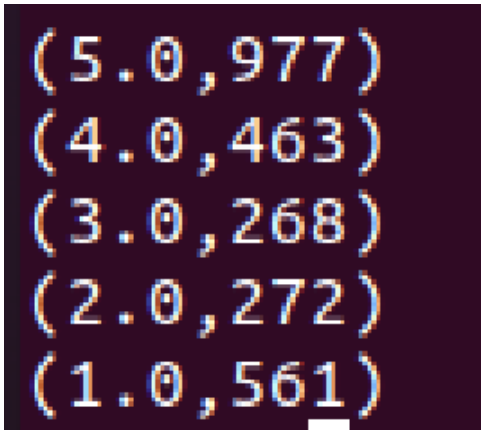
Le due parole valutate solo le due più frequenti ottentute dalla query (2), ossia 'food' e 'time. Il codice in Pig Latin utilizzato è il seguente:

```
R = LOAD '/media/sf_Challenge/yelp_academic_dataset_review.json' USING JsonLoader
    ↪ ('review_id: chararray, user_id: chararray, business_id: chararray, stars:
    ↪ float, useful: chararray, funny: chararray, cool: chararray, text:
    ↪ chararray, date: chararray');

Review_cut = LIMIT R 10000;
P = FOREACH Review_cut GENERATE FLATTEN(TOKENIZE(text)),stars;
```



(a) Risultato per la parola 'food'



(b) Risultato per la parola 'time'

Figura 3: Risultato Query 2

```
word = FILTER P BY $0=='food'; // word = FILTER P BY $0=='time';
G = GROUP word BY $1;
Ris = FOREACH G GENERATE $0, COUNT($1);
unmix = ORDER Ris BY $0 DESC;
DUMP unmix;
```

Listing 5: Codice Seconda query

Come è possibile dedurre da risultati ottenuti, la parola *food* è associata maggiormente a recensioni positive, mentre dalla parola *time*, nonostante le recensioni positive siano maggiori, non si può dedurre un comportamento in particolare, dunque è possibile etichettare la parola come neutra.

4.3 Query 3

Attraverso questa query si vuole ottenere un conteggio generale sul numero di recensioni per ogni categoria di valutazione in stelle.

```
R = LOAD '/media/sf_Challenge/yelp_academic_dataset_review.json' USING JsonLoader
    ↪ ('review_id: chararray, user_id: chararray, business_id: chararray, stars:
    ↪ float, useful: chararray, funny: chararray, cool: chararray, text:
    ↪ chararray, date: chararray');

Review_cut = LIMIT R 10000;
grouped_r= GROUP Review_cut BY stars;
cntd = FOREACH grouped_r GENERATE $0, COUNT($1);
STORE cntd INTO '/media/sf_Challenge/outputstars' USING PigStorage (',');
```

Listing 6: Codice Prima Query

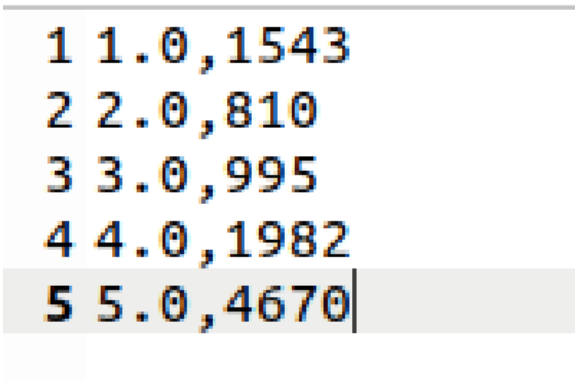


Figura 4: Risultato Query 3

Come è possibile dedurre dai risultati ottenuti, le recensioni più frequenti sono quelle che danno 1, 4 e 5 stelle. Ciò fa comprendere un particolare comportamento degli utenti, infatti quest'ultimi sono soliti dare recensioni o estremamente negative oppure estremamente positive, mentre sono meno frequenti delle recensioni "neutre".

4.4 Query 4

Per elaborare il risultato della query riportata qui di seguito è stato utilizzato il file Business.json. Si vogliono ricavare le città che presentano il maggior numero di recensioni, in modo tale da avere una panoramica geografica dell'utilizzo dell'applicazione Yelp, Quindi sono state *ordinate le città per numero di recensioni in ordine decrescente.*

```
BS = LOAD '/media/sf_Challenge/yelp_academic_dataset_business.json' USING
    ↪ JsonLoader('business_id: chararray, name: chararray, address: chararray,
    ↪ city: chararray, state: chararray, postal_code: chararray, latitude: float,
    ↪  longitude: float, stars: float, review_count: int, is_open: int,
    ↪ attributes: chararray, categories: chararray, hours:chararray');

grouped = GROUP BS BY city;
conteggio = FOREACH grouped GENERATE $0, COUNT($1);
unmix = ORDER conteggio BY $1 DESC, $0 ASC;
STORE unmix INTO '/media/sf_Challenge/output' USING PigStorage (',');
```

Listing 7: Codice Prima Query

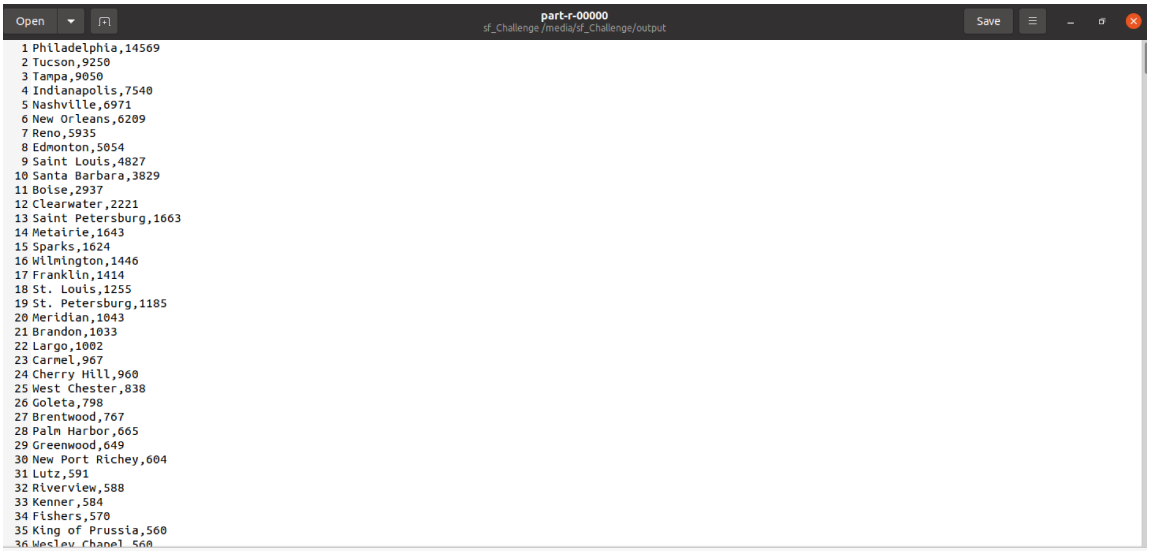


Figura 5: Risultato Query 1

5 Modalità Hadoop

Come già accennato nel paragrafo precedente, per ottenere i risultati descritti in precedenza è stata impiegata la modalità locale offerta da Apache Pig. Tuttavia, ogni query sarebbe potuta essere implementata anche nel caso della modalità Hadoop con opportune modifiche. La prima consiste nella modifica al codice per avviare Pig da terminale Linux. In particolare, basta eseguire il semplice comando:

```
./pig
```

Listing 8: Hadoop Mode

La seconda principale differenza consiste nelle operazioni di LOAD e STORE, in quanto i file non sono più locali all'interno del dispositivo, ma sono letti e scritti da e sui nodi di Hadoop. Dunque, per salvare un file locale su un nodo Hadoop sarà necessario il seguente comando:

```
hdfs dfs -put /local-file-path /hdfs-file-path
```

Successivamente, quando si scriverà la query dalla shell *Grunt* (quella utilizzata per inserire le istruzioni in Pig Latin), le operazioni di load e store dovranno essere scritte nel seguente modo:

```
grunt> VariableName = LOAD 'hdfs://localhost:9000/file-path' USING PigStorage
    ↪ (',');
grunt> STORE VariableName INTO ' hdfs://localhost:9000/pig_Output/ ' USING
    ↪ PigStorage(',');
```

Una volta fatto ciò è possibile accedere al file contenente il risultato delle elaborazioni tramite shell inserendo il seguente comando:

```
hdfs dfs -cat 'hdfs://localhost:9000/pig_Output/part-m-00000'
```

6 Considerazioni finali

Nel corso di questo elaborato è stato possibile comprendere quanto il linguaggio Pig Latin sia performante, in quanto permette di eseguire operazioni complesse in poche e semplici righe di codice. Di converso, uno svantaggio che è stato riscontrato consiste nella difficoltà di lavorare con file Json *schemaless*, in quanto JsonLoader(), ossia la funzione built-in offerta da Pig per lavorare con file in formato JSON, necessita in ingresso di uno schema della struttura del file JSON. Una soluzione a tale problema, anche se non direttamente connessa al presente elaborato, consiste nell'importare alcune librerie del seguente progetto GitHub: `https://github.com/twitter/elephant-bird` ed, in particolare nell'utilizzo della seguente funzione:

```
com.twitter.elephantbird.pig.load.JsonLoader();
```