Guida operativa

Federico Luzzi (DS), Marco Peracchi (DS), Christian Uccheddu (DS), Gabriele Celeri (TTC)

Di seguito la guida operativa per eseguire il codice e replicare i risultati ottenuti:

Presa dati Dicembre-Gennaio

Eseguire il file che esegue richieste ogni 30 minuti e salva i dati in formato csv.

`scraper/scraper_csv.py`

Il jupyter-notebook serve a selezionare prese dati ogni 6 ore, e si trova all'interno della cartella *clean_store_data*.

`clean_jen_video/fix_jen_json.ipynb`

Per trasformare i dati da formato csv a json.

`csv_to_json/main.py`

Lo script ha bisogno del parametro:

- -d "directory_dei_dati"

In tal modo si indica la directory dove si trovano i dati da trasformare. Per caricare i json su MongoDB avviare lo script:

`json_to_mongo(windows)/json_to_mongo.py`

A questo script vanno forniti i seguenti parametri:

- -d "directory dei dati"
- -u "utente mongo"
- -p "password utente"
- -port "porta in cui è attivo mongoDB"
- -db "nome del database mongo"
- -c "collection in cui vengono inseriti i dati"

Presa dati Marzo-Maggio

Aprire il servizio MongoDB, zookeeper e conseguentemente Kafka. Eseguire in due terminali separati secondo il seguente ordine gli script:

1. `scraper/scraper_consumer.py`
2. `scraper/scraper_producer.py`

Ogni 6 ore il *producer* effettua le richieste dati che invia poi al canale Kafka letto dal *consumer* che si occupa della pulizia e immagazzinamento dei dati.

Presa dati Covid

Scaricare i dati in formato csv da OurWorldInData ed eseguire il codice:

`covid/cleaner.py`

Questo script permette di eseguire una pulizia dei dati in modo da renderli integrabili con i json dei video precedenti.

Integrazione dei dati

Per ottenere l'integrazione tra i dati Covid e i dati di Youtube bisogna eseguire il seguente script:

`json_to_mongo(windows)/merge_to_mongo.py`

A questo script vanno forniti i seguenti parametri:

- -d "directory dei dati"
- -u "utente mongo"
- -p "password utente"
- -port "porta in cui è attivo mongodb"
- -db "ome del database mongo"
- -c "collection in cui vengono inseriti i dati"
- -e "path dove c'è file dell'espressione regolare"

I dati vengono quindi integrati sulla data e il paese e successivamente caricati su Mongo.

Query mongo

La seconda domanda di ricerca richiede di distinguere quali video riguardino il coronavirus o meno. L'espressione regolare seguente analizza tag e titoli per verificare se è presente una parola riferita alla pandemia.

```
/(corona|covid|virus|pandemi[aec]|epidemi[aec]|tampon[ei]*|sierologico|mascherin[ae]|코로나 바
이러스|fase\s*(2|due)|iorestoacasa|stayathome|lockdown|[qc]uar[ae]nt[äae]i*n
[ea]|कोरोनावाइरस|कोरोना|महामारी|massisolation|distanziamento\s*sociale|social\s*distancing|감염병 세계적 유행
|パンデミック|コロナウイルス|सर्वव्यापी महामारी|सार्वत्रिक महामारी|пандемия|коронавирус|social\s*distancing|
distanciamiento\s*social|코로나|कोविड|कोरोना|vaccin[oe]*|isolamento|intensiv[ao]|assembrament[io]|
guant[oi]|dpi|disinfettante|swabs|emergenza|emergency|droplets*|aerosol|isolation|intensive
\s*care|crowd|gloves*|disinfectant|감염병 유행|완충기|마스크|나는 집에있어|폐쇄|사회적 거리두기|백신|모임|비상 사
태|비말|봄 혈증|écouvillon|masques*|restealamaison|confin[ae]mento*|distanciation\s*sociale|
soins\s*intensifs|rassemblements|désinfectant|urgence|gouttelettes|飛沫|タンポン|マスケリン|封鎖|
人混みを避ける|ワクチン|隔離|集会|集中治療|緊急|कर्फ़्यू|फाहे|मास्क|लॉकडाउन|सोशल डिस्टन्सिंग|टीका|गहन देखभाल|समारोहों|आपातकालीन|
gotas|cotonetes|m[áa]scaras|ficoemcasa|vac[iu]na|reuni[õo]n*es|emerg[êe]ncia|капли|тампоны|
маски|карантин|социальное\s*дистанцирование|вакцина|интенсивная\s*терапия|сходы|
чрезвычайное\s*происшествие|hisopos|mequedoencasa|cierre|Tröpfchen|Tupfer|Masken|
bleibezuHause|Ausgangssperre|soziale\s*Distanzierung|Impfstoff|Intensivstation|
Versammlungen|Notfall|건강\s*격리|検疫|संगरोध|[кK]арантин|hand|man[io]s*|소유|手|[Pp]уки|Hände|
mãos)/i
```

Figura 1: Regular expression usata

Per applicare la query sui documenti si possono eseguire singolarmente i comandi in una shell mongo, come viene descritto qui di seguito, oppure è possibile chiamare una funzione che esegue automaticamente tutte e tre le query.

Eseguire ora i seguenti comandi all'interno della shell di MongoDB.

Creare due nuovi campi chiamati **covid_title** e **covid_tags** inizializzati entrambi a **False**

```
db.video_merge.update({},{$set : {covid_tags : false,
                        covid_title : false}},
                        {multi : true})
```

Eseguire le seguenti due query che controllano se l'espressione regolare è presente nel campo title o in uno dei tag per ogni video.

```
db.video_merge.update({tags : {$in : [REGEX]}},
                        {$set : {covid_tags: true}},
                        {multi : true})
```

```
db.video_merge.update({title : {$in : [REGEX]}},
                      {$set : {covid_title: true}},
                      {multi : true})
```

È possibile eseguire da file le precedenti espressioni, attraverso lo script *query_ covid.py*. Esso viene chiamato come funzione esterna *compute_ covid* durante l'integrazione nello script *merge_ to_ mongo.py* automaticamente.

Sharding

Lo sharding dei documenti di Mongo è stato eseguito in locale, quindi utilizzando *localhost* come host. All'interno della cartella *sharding* è possibile visualizzare le cartelle contenenti i vari file di configurazione per tutti i componenti.

- **configsvr** Sono tre istanze di *mongod*, configurate come replica set. Il *config server* conosce dove ogni dato è allocato dei vari shard, quindi è importante configurarlo come replica-set, così che in caso di guasti non si perdano le informazioni.
- **router** È un'istanza di *mongos*. Per interrogare i vari shard è necessario interfacciarsi con essa.
- **shard** Sono tre istanze di *mongod*. Ogni shard è configurato in replica-set, e i dati vengono suddivisi nei vari shard.

All'interno di ogni cartella sono presenti i vari file di configurazione, dove deve essere specificato il percorso della cartella *data* di ogni istanza. Inoltre bisogna precisare che, essendo configurato tutto in una macchina locale, è importante utilizzare porte differenti per ogni istanza.
Per prima cosa inizializzare i replica-set, e successivamente avviarli come shard server o come config server. Una volta collegati quest'ultimi al router è necessario caricare i dati su uno degli shard. Prima di effettuare la procedura va specificata la *shard key*, e va anche specificato il metodo di suddivisione dei dati, se *hashed* o *ranged*. In questo lavoro è stato utilizzato il primo metodo.
Le collezioni che abbiamo shardato sono state due, una contenente i dati di marzo-maggio integrati con i dati Covid, e l'altra contenente i video di dicembre-gennaio e marzo-maggio. I dati vengono caricati attraverso il router sugli shard, specificando la shard key, e automaticamente suddivisi.

Per ulteriori informazioni sui comandi leggere l'appendice.

Appendice

Sharding

Creazione Config Server

Vengono configurati i tre config server:

```
mongod --config <file.cfg> --configsvr
```

Eseguire il comando sul config server PRINCIPALE, vengono indicati i tre membri del replica set, il quale nome sarà rs0:

```
rs.initiate({ _id: "rs0",
configsvr: true, members: [{ _id : 0, host : "localhost:27019" },
{ _id : 1, host : "localhost:27020" },
{ _id : 2, host : "localhost:27021" }
]})
```

Creazione Shard Server

Aprire tutte le istanze di mongod dei vari shard.

```
mongod --config <file.cfg> --shardsvr
```

Successivamente per ogni gruppo replica-set accedere ad uno ed inizializzare il gruppo:

- Primo gruppo

```
rs.initiate(
{_id : "rs1",
members: [
{ _id : 0, host : "localhost:27022" },
{ _id : 1, host : "localhost:27023" },
{ _id : 2, host : "localhost:27024" }
]
}
)
```

- Secondo gruppo

```
rs.initiate(
{_id : "rs2",
members: [
{ _id : 0, host : "localhost:27025" },
{ _id : 1, host : "localhost:27026" },
{ _id : 2, host : "localhost:27027" }
]
}
)
```

- Terzo gruppo

```
rs.initiate(
{_id : "rs3",
members: [
{ _id : 0, host : "localhost:27028" },
{ _id : 1, host : "localhost:27029" },
{ _id : 2, host : "localhost:27030" }
]
}
)
```

Router

Dopo aver inizializzato gli shard e i config server aprire il router come istanza di mongos:

```
mongos --config <file.cfg>
```

Dopo aver avviato il router, connettersi ad esso attraverso il localhost e la porta ad esso associata mediante il comando:

```
mongo --host <router>
```

Ora è possibile aggiungere i vari shard mediante il comando *sh.addShard()* al router, inserendo tra le parentesi l'indirizzo di collegamento allo shard. Nel nostro caso, avendo impostato come replica-set ogni shard è necessario utilizzare:

- Primo shard

```
sh.addShard("rs1/localhost:27022,localhost:27023,localhost:27024")
```

- Secondo shard

```
sh.addShard("rs2/localhost:27025,localhost:27026,localhost:27027")
```

- Terzo shard

```
sh.addShard("rs3/localhost:27028,localhost:27029,localhost:27030")
```

Adesso è possibile caricare i dati mediante il router mongos e applicare lo shard, utilizzando come chiave il paese. Il comando da utilizzare è:

```
sh.shardCollection("YT_data.video_all", {country_name : "hashed"})
```

La collezione risulta ora distribuita su tre shard, tutti configurati come replica-set.

Versione Software

Le versioni utilizzate sono:

- **Zookeeper** 3.4.10
- **Apachi Kafka** 2.3.0
- **MongoDB** 4.2
- **Python** 3.6
- **Tableau** 2019.4.7
- **GitHub** 2.20.1 Link al repository