Pontifícia Universidade Católica de Minas Gerais
Pós-graduação Lato Sensu – 2° Semestre 2016
Ciência de Dados e Big Data
Bancos de dados não relacionais

Prof. Gabriel Campos

Willian Hofner

# Trabalho Prático

● **Coletar informações de redes sociais e armazenar ~1M de dados.**

O tema escolhido foi o atentado a feira de natal em Berlim em Dez de 2016.

Ferramentas utilizadas para a análise:

- Ultrabook Inspirion Intel® Core™ i5-3337U CPU @ 1.80GHz

- Memória 6 GB de RAM

- Sistema Operacional de 64 bits Windows 10 Enterprise

- Python 2.7.10

- Mongodb 3.4

- Tweepy

Foi definido como chave de busca os termos:

['ARVBerlin', 'Terrorism', 'Terrorismo', 'Berlin']

Apesar da coleta ser de ~1M de twiters, houve problemas de paradas na captura dos dados, apenas "count" : 285203 twites, foram capturados, devido a problemas de internet, banda da rede com muito trafego, etc.

- Foi desenvolvido uma aplicação em Python para a Listener de nome “twitter_streaming.py”.

Retorno na tela do shell do mongodb após executar o comando do mapReduce:

```
>db.tweets.mapReduce(map, reduce, {out: "word_freq"})
{
    "result" : "word_freq",
    "timeMillis" : 246504,
    "counts" : {
```

```
        "input" : 285203,
        "emit" : 5436611,
        "reduce" : 755711,
        "output" : 254641
    },
    "ok" : 1
}
```

Abaixo o display na shell do mongodb da estatística coletada:

```
> db.tweets.stats()
{
    "ns" : "twitter_db.tweets",
    "size" : 70356494,
    "count" : 285203,
    "avgObjSize" : 246,
    "storageSize" : 40816640,
    "capped" : false,
    "wiredTiger" : {
        "metadata" : {
            "formatVersion" : 1
        },
...
```

- **Extrair informações do tipo :**
  - **Termos mais frequentes**

- Foram feitas as funções Map e Reduce com o nome “termos_mais_frequentes.js” para a extração dos termos mais frequentes.

Retorno da shell do mongodb após execução do mapReduce dos termos mais frequentes:

```
> db.tweets.mapReduce(map, reduce, {out: "word_freq"})
{
    "result" : "word_freq",
    "timeMillis" : 246504,
    "counts" : {
        "input" : 285203,
        "emit" : 5436611,
        "reduce" : 755711,
        "output" : 254641
    },
    "ok" : 1
}
```

Abaixo o display na shell do mongodb da “db.word_freq” dos termos mais frequentes:

```
> db.word_freq.find().sort({value : -1});
{ "_id" : "radio", "value" : 185 }
{ "_id" : "Terroristen", "value" : 177 }
{ "_id" : "Berlin.", "value" : 175 }
{ "_id" : "sein", "value" : 175 }
{ "_id" : "Jack", "value" : 173 }
{ "_id" : "Y", "value" : 173 }
{ "_id" : "come", "value" : 173 }
{ "_id" : "put", "value" : 173 }
{ "_id" : "Great", "value" : 172 }
{ "_id" : "wenn", "value" : 172 }
{ "_id" : "yet", "value" : 172 }
{ "_id" : "Alemania", "value" : 171 }
{ "_id" : "One", "value" : 171 }
{ "_id" : "Thanks", "value" : 171 }
{ "_id" : "Al", "value" : 170 }
{ "_id" : "fue", "value" : 170 }
{ "_id" : "give", "value" : 170 }
{ "_id" : "long", "value" : 170 }
{ "_id" : "safe", "value" : 170 }
{ "_id" : "thought", "value" : 170 }
...
```

- **Volume x dia**

- Foi feita a função de agregação “volume_dia.js”.

Abaixo o display na shell do mongodb de agregação do volume por dia:

```
>db.tweets.aggregate([{$match:{"created_at":/2016/}} , {$group: { "_id":"$data", "total":{$sum:1} } }  ])
{ "_id" : "Thu Dec 22", "total" : 153833 }
{ "_id" : "Fri Dec 23", "total" : 27123 }
{ "_id" : "Wed Dec 21", "total" : 104247 }
```

- **Volume x hora dia**

- Foi feita a função de agregação “volume_hora_dia.js”.

Abaixo o display na shell do mongodb de agregação do volume por dia:

```
> db.tweets.aggregate([{$match:{"created_at":/2016/}} , {$group: { "_id":"$hora", "total":{$sum:1} } } ])
{ "_id" : "Fri Dec 23 14", "total" : 7876 }
{ "_id" : "Fri Dec 23 00", "total" : 191 }
{ "_id" : "Thu Dec 22 22", "total" : 345 }
{ "_id" : "Thu Dec 22 20", "total" : 15512 }
{ "_id" : "Thu Dec 22 16", "total" : 7115 }
{ "_id" : "Thu Dec 22 21", "total" : 4145 }
{ "_id" : "Thu Dec 22 14", "total" : 12947 }
{ "_id" : "Thu Dec 22 11", "total" : 4858 }
{ "_id" : "Fri Dec 23 13", "total" : 19056 }
{ "_id" : "Thu Dec 22 08", "total" : 5707 }
{ "_id" : "Thu Dec 22 09", "total" : 2142 }
{ "_id" : "Thu Dec 22 12", "total" : 1381 }
{ "_id" : "Thu Dec 22 02", "total" : 11141 }
{ "_id" : "Wed Dec 21 23", "total" : 8651 }
{ "_id" : "Thu Dec 22 15", "total" : 15902 }
{ "_id" : "Wed Dec 21 21", "total" : 14806 }
{ "_id" : "Thu Dec 22 18", "total" : 3103 }
{ "_id" : "Wed Dec 21 20", "total" : 16226 }
{ "_id" : "Thu Dec 22 10", "total" : 13811 }
{ "_id" : "Thu Dec 22 00", "total" : 11705 }
```

Conclusão:

O desafio maior deste trabalho foi não conhecer a captura de listener do Tweepy. Foram gastos várias horas de estudo para compreensão do código do Tweepy, Python e Mongodb.

Apesar do listener cair algumas vezes conseguiu-se a captura de "count" : 285203 twites, e foi com esta massa de dados que foi feito a análise do atentado a feira de natal em Berlim do que foi pedido.

Bibliografia:


1. Mongodb Manual. Consulta Map-Reduce e Aggregate. Disponível em: https://docs.mongodb.com/manual/
2. Regular Expressions In Javascript. Disponível em: http://evolt.org/regexp_in_javascript
3. Tweepy Documentation. Disponível em: http://docs.tweepy.org/en/v3.5.0/
4. Python Software Foundation. Disponível em: https://www.python.org/
5. Twitter Developer Documentation. Disponível em: https://dev.twitter.com/
6. API Documentation for MongoDB Drivers. Disponível em: https://api.mongodb.com/