PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Curso de Especialização em Ciência de Dados e Big Data

Banco de dados não relacionais

Trabalho prático de extração de dados

Pedro Rocha Goecking Silva

BELO HORIZONTE

DEZEMBRO DE 2016

SUMÁRIO

1 CONTEXTUALIZAÇÃO

O objetivo deste capítulo é introduzir e explicar o problema, a necessidade para este trabalho prático e, ao mesmo tempo, apresentar as soluções utilizadas, explicando suas funcionalidades.

1.1 Enunciado

- Coletar informações de redes sociais ou importar dados externos e armazenar ~1M de dadis em banco NoSQL.
- Extrair informações do tipo:
  - Termos mais frequentes
  - Volume x dia
  - Volume x Hora do dia

1.2 Python

Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Possui dependências que auxiliam desenvolvedores com soluções prontas. Essas dependências, ou pacotes, são gerenciadas pelo *pip*, um sistema gerenciador de pacotes utilizado para instalar, remover e atualizar os referidos pacotes. A maioria dos pacotes podem ser encontrados no Python Package Index (PyPI).

O Python foi escolhido para este cenário por ser uma linguagem de fácil utilização, propicia para a exploração de dados, disponibilizando soluções prontas através de pacotes e extensões.

1.2.1 Tweepy

O Tweepy é o pacote responsável por abstrair uma camada entre o código desenvolvido e a API do Twitter. É restrito à função descrita anteriormente, mas tem grande utilidade por possibilitar ao desenvolvedor capturar tuítes do mundo inteiro em tempo real baseado apenas num dicionário de palavras-chave. Sua instalação foi feita por meio do comando apresentado abaixo:

```
nosql@virtualbox:~/Aulas$ pip install tweepy
```

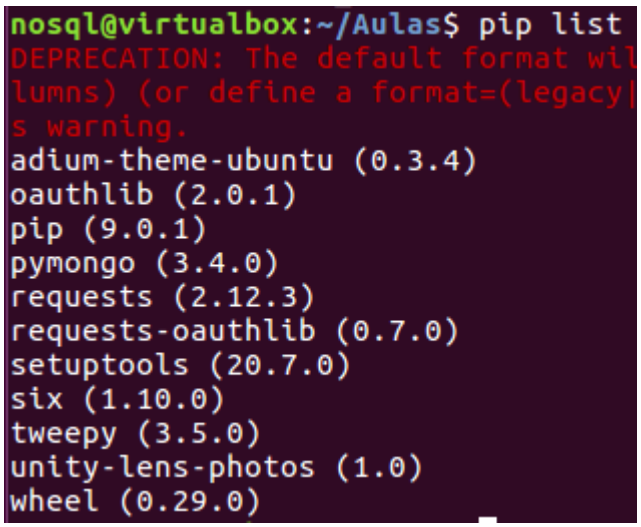
1.2.2 PyMongo

De acordo com o site do MongoDB, PyMongo é uma distribuição Python contendo ferramentas para se trabalhar com MongoDB, sendo recomendado seu uso para manipulação de dados envolvendo MongoDB a partir do Python.

Através deste driver ocorre a comunicação da aplicação com o banco não relacional referido. Sua instalação se deu da mesma forma como descrita no Tweepy:

```
nosql@virtualbox:~/Aulas$ pip install pymongo
```

A imagem abaixo comprova a instalação dos pacotes descritos anteriormente:



2 DESENVOLVIMENTO

Após a criação da conta no Twitter e da geração do app através do link https://apps.twitter.com/, bem como os tokens e Keys necessários, o código apresentado a seguir foi utilizado para captura em tempo real de tuítes, utilizando como pano de fundo o atual cenário de tensão internacional entre extremistas radicais islâmicos apoiados por interesses de grandes nações e o ocidente. As palavras chaves escolhidas foram "berlin", "aleppo", "terrorism", "muslim", "attack", "russian", "syria", "turkey", "usa", "united states", "terrorist", "death", “dead", "die", "army" e "russia", portanto, optou-se por capturar tuítes em inglês.

2.1 Código e explicações

```python
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
from pymongo import MongoClient

ckey = 'Et95tH8EG5jP3IUc2KXSA1uik'
csecret = 'jnuA0ser9t37WA7eYavPr1ud8KF9oYfOop1qKcEHNpFyie0iCr'
atoken = '808007428663087104-3bqlBZJsiyiqHXP4el0LJuGzwdhPSgS'
asecret = 'FTZxWjEFRvHAyr3kK5f2o882o7mHZwZTp33cfSFEVj4qM'

class listener(StreamListener):
    def on_data(self, data):
        tweet=json.loads(data)
        created_at = tweet["created_at"]
        id_str = tweet["id_str"]
        text = tweet["text"]
        obj = {"created_at":created_at,
        "id_str":id_str,
        "text":text,}
        tweetind=collection.insert_one(obj).inserted_id
        print obj
        return True
    def on_error(self, status):
        print status


client = MongoClient()
client = MongoClient('localhost', 27017)
db = client.nosqlclass
collection = db.test_collection

auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
twitterStream = Stream(auth, listener())
twitterStream.filter(track=["berlin","aleppo","terrorism","muslim","attack","russian","syria","turkey",
                            "usa","united states","terrorist","death", "dead","die","army","twitter","russia"])
```

A imagem anterior mostra todo o código utilizado. Seguidamente, é apresentada a explicação do código Python.

O trecho destacado abaixo contém a importação dos pacotes e suas dependências. Através destes recursos é feita a autenticação (OAuthHandler), escuta e captura de tuítes (Stream e StreamListener) e a comunicação com o banco NoSQL Mongo (MongoClient).

```python
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
from pymongo import MongoClient
```

O trecho destacado abaixo contém os parâmetros gerados no Twitter App. Com essas Keys e Tokens a aplicação tem acesso aos tuítes.

```python
ckey = 'Et95tH8EG5jP3IUc2KXSA1uik'
csecret = 'jnuA0ser9t37WA7eYavPr1ud8KF9oYfOop1qKcEHNpFyie0iCr'
atoken = '808007428663087104-3bqlBZJsiyiqHXP4el0LJuGzwdhPSgS'
asecret = 'FTZxWjEFRvHAyr3kK5f2o882o7mHZwZTp33cfSFEVj4qM'
```

O trecho destacado abaixo contém a classe que faz a escuta e captura de tuítes efetivamente. Na função on_data, existem algumas variáveis que armazenam dados relacionados ao tuíte, além do texto, como o ID e a hora de criação (postagem) do tuíte na rede. Todos esses dados são concatenados dentro de obj que envia para o collection definido.

A função on_error retorna o status quando houver alguma exceção.

```
class listener(StreamListener):
    def on_data(self, data):
        tweet=json.loads(data)
        created_at = tweet["created_at"]
        id_str = tweet["id_str"]
        text = tweet["text"]
        obj = {"created_at":created_at,
        "id_str":id_str,
        "text":text,}
        tweetind=collection.insert_one(obj).inserted_id
        print obj
        return True
    def on_error(self, status):
        print status
```

O trecho destacado abaixo contém a conexão com o MongoDB. O servidor é local, a porta é 27017, o nome do banco é nosqlclass e o collection para onde os tuítes são enviados é o test_collection.

```
client = MongoClient()
client = MongoClient('localhost', 27017)
db = client.nosqlclass
collection = db.test_collection
```

O trecho destacado abaixo contém a invocação da aplicação, onde são passados os tokens, Keys e palavras-chaves definidas.

```
auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
twitterStream = Stream(auth, listener())
twitterStream.filter(track=["berlin","aleppo","terrorism","muslim","attack","russian","syria","turkey",
```

3 RESULTADOS

Este capítulo tem por objetivo apresentar os resultados da captura dos tuítes. Primeiramente é exibido a quantidade de tuítes por palavras-chaves previamente selecionadas.

TERMO	QUANTIDADE
attack	40.925
die	15.125
dead	11.125
terrorist	10.950
death	6.550
usa	5.925
terrorism	5.725
russia	3.800
berlin	3.775
russian	3.475
muslim	3.350
twitter	3.200
turkey	1.950
army	1.000
aleppo	750
syria	575
united states	50

Em relação ao percentual de captura por dia, temos o seguinte resultado:

DIA	%
20/12	21,8%
21/12	28,0%
22/12	6,2%
23/12	44,1%

E, finalmente, em relação à hora, temos o seguinte resultado:

HORA	%
00	13,51%
01	13,62%
02	4,29%
04	0,18%
08	1,23%
09	1,59%
10	0,35%
11	14,56%
12	1,88%
13	25,31%
14	2,58%
15	1,17%
16	0,70%
17	2,88%
18	0,29%
19	10,63%
20	5,05%
21	0,06%
23	0,12%

4 CONCLUSÕES

4.1 Dificuldades

Durante o processo de desenvolvimento do trabalho, observou-se grande dificuldade em relação à satisfação do requisito proposto da captura de 1 milhão de tuítes. O que ocorria após poucos segundos de cada tentativa de execução do código era o erro raise exception, conforme a imagem abaixo apresenta:

```
Traceback (most recent call last):
  File "nosql/trabalho_pratico/twitter_streaming.py", line 36, in <module>
    twitterStream.filter(track=["berlin","aleppo","terrorism","muslim","attack","russian","syria","tu
rkey", "usa","united states","terrorist","death", "dead","die","army","twitter","russia"])
  File "/home/nosql/.local/lib/python2.7/site-packages/tweepy/streaming.py", line 445, in filter
    self._start(async)
  File "/home/nosql/.local/lib/python2.7/site-packages/tweepy/streaming.py", line 361, in _start
    self._run()
  File "/home/nosql/.local/lib/python2.7/site-packages/tweepy/streaming.py", line 294, in _run
    raise exception
KeyError: 'created_at'
```

Com base na troca de informações entre colegas, pode-se aferir duas causas para o problema. O primeiro é o tráfego de rede. Nos horários de pico da

internet, de 19h às 22h (horário comprovadamente de maior uso da internet no Brasil), a execução durava menos tempo do que em horários alternativos, como o período da tarde e madrugada. A outra possível causa é o bloqueio por parte da API do Twitter para captura de dados. Outros sites conhecidos possuem verificadores de crawling, como o Google, e fazem o bloqueio temporário do usuário. Possivelmente a API do Twitter limita o acesso aos tuítes, o que, no meu caso, forçou a execução constante do código-fonte.

4.2 Aprendizado

Apesar de ter sido desafiadora a tarefa de implementar um código que capturasse dados não estruturados da Web, o grande aprendizado que pode-se absorver deste trabalho prático são as possibilidades de uso do dado e a forma como ele é gerenciado. A utilização de um novo paradigma para banco de dados ampliou os horizontes, fazendo com que novas possibilidades passassem a ser consideradas em análise de dados.

5 BIBLIOGRAFIA

- https://pt.wikipedia.org/wiki/Python
  Acesso em: <23/12/2016> às 02:57
- https://fbormann.wordpress.com/2015/09/03/drop-3-tweepy-e-pandas-primeiras-impressoes/
  Acesso em: <23/12/2016> às 02:57
- https://docs.mongodb.com/getting-started/python/client/
  Acesso em: <23/12/2016> às 02:57
- https://www.linkedin.com/pulse/collecting-twitter-stream-using-python-mongodb-shailendra
  Acesso em: <23/12/2016> às 02:57
- https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/?completed=/mysql-live-database-example-streaming-data/
  Acesso em: <23/12/2016> às 02:57
- https://apps.twitter.com/
  Acesso em: <23/12/2016> às 02:57

- http://www.ebricksdigital.com.br/pesquisas-de-mercado/pesquisa-aponta-o-horario-nobre-da-internet-no-brasil/
  Acesso em: <23/12/2016> às 02:57