DSA-Python-Cap06-09-Stream de Dados do Twitter com MongoDB, Pandas e Scikit Learn

January 2, 2021

```
In [ ]: # Versão da Linguagem Python
        from platform import python_version
        print('Versão da Linguagem Python Usada Neste Jupyter Notebook:', python_version())
```

0.1 Stream de Dados do Twitter com MongoDB, Pandas e Scikit Learn

0.2 Preparando a Conexão com o Twitter

```
In [ ]: # Instala o pacote tweepy
        !pip install tweepy
```

```
In [ ]: # Importando os módulos Tweepy, Datetime e Json
        from tweepy.streaming import StreamListener
        from tweepy import OAuthHandler
        from tweepy import Stream
        from datetime import datetime
        import json
```

Veja no manual em pdf como criar sua API no Twitter e configure as suas chaves abaixo.

```
In [ ]: # Adicione aqui sua Consumer Key
        consumer_key = "JnX7T8LFrRzJzPeEQi5YXM02J"
```

```
In [ ]: # Adicione aqui sua Consumer Secret
        consumer_secret = "fCU8drm7SGOoRfWcPYmPSPmUJbhnT94R6v7unyB7qFv1IDb3vd"
```

```
In [ ]: # Adicione aqui seu Access Token
        access_token = "AAAAAAAAAAAAAAAAAAAAAK7LLAEAAAAAnykLsiD7%2FVM2dEM1d3TAeFPjjOg%3DRgu3ull
```

```
In [ ]: # Adicione aqui seu Access Token Secret
        access_token_secret = "LUCVaG43pAJEHqrWkwBS66M6xkF27zOATyUAoU8zzYntX"
```

```
In [ ]: # Criando as chaves de autenticação
        auth = OAuthHandler("JnX7T8LFrRzJzPeEQi5YXM02J", "fCU8drm7SGOoRfWcPYmPSPmUJbhnT94R6v7u
```

```
In [ ]: auth.set_access_token(access_token, access_token_secret)
```

```
In [ ]: # Criando uma classe para capturar os stream de dados do Twitter e
        # armazenar no MongoDB
        class MyListener(StreamListener):
            def on_data(self, dados):
                tweet = json.loads(dados)
                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 = col.insert_one(obj).inserted_id
                print (obj)
                return True
```

```
In [ ]: # Criando o objeto mylistener
        mylistener = MyListener()
```

```
In [ ]: # Criando o objeto mystream
        mystream = Stream(auth, listener = mylistener)
```

0.3 Preparando a Conexão com o MongoDB

```
In [ ]: # Importando do PyMongo o módulo MongoClient
        from pymongo import MongoClient
```

```
In [ ]: # Criando a conexão ao MongoDB
        client = MongoClient('localhost', 27017)
```

```
In [ ]: # Criando o banco de dados twitterdb
        db = client.twitterdb
```

```
In [ ]: # Criando a collection "col"
        col = db.tweets
```

```
In [ ]: # Criando uma lista de palavras chave para buscar nos Tweets
        keywords = ['Big Data', 'Python', 'Data Mining', 'Data Science']
```

0.4 Coletando os Tweets

```
In [ ]: # Iniciando o filtro e gravando os tweets no MongoDB
        mystream.filter(track=keywords)
```

0.5 –> Pressione o botão Stop na barra de ferramentas para encerrar a captura dos Tweets

0.6 Consultando os Dados no MongoDB

```
In [ ]: mystream.disconnect()
```

```
In [ ]: # Verificando um documento no collection
        col.find_one()
```

0.7 Análise de Dados com Pandas e Scikit-Learn

In [ ]:
```
# criando um dataset com dados retornados do MongoDB
dataset = [{"created_at": item["created_at"], "text": item["text"],} for item in col.fi
```

In [ ]:
```
# Importando o módulo Pandas para trabalhar com datasets em Python
import pandas as pd
pd.__version__
```

In [ ]:
```
# Criando um dataframe a partir do dataset
df = pd.DataFrame(dataset)
```

In [ ]:
```
# Imprimindo o dataframe
df
```

In [ ]:
```
# Importando o módulo Scikit Learn
from sklearn.feature_extraction.text import CountVectorizer
```

In [ ]:
```
import sklearn
sklearn.__version__
```

In [ ]:
```
# Usando o método CountVectorizer para criar uma matriz de documentos
cv = CountVectorizer()
count_matrix = cv.fit_transform(df.text)
```

In [ ]:
```
# Contando o número de ocorrências das principais palavras em nosso dataset
word_count = pd.DataFrame(cv.get_feature_names(), columns=["word"])
word_count["count"] = count_matrix.sum(axis=0).tolist()[0]
word_count = word_count.sort_values("count", ascending=False).reset_index(drop=True)
word_count[:50]
```

In [ ]:
```
# Fim
```

In [ ]: