Usando Elasticsearch com Python

October 27, 2020

1 Utilizando Elasticsearch com Python

1.1 Importando o pacote cliente do Elasticsearch

Para instalar o pacote do Elasticsearch, use:

```
pip install elasticsearch
```

[1]:
```
import json

import pandas as pd

%matplotlib inline

import matplotlib.pyplot as plt
```

[2]:
```
import elasticsearch

client = elasticsearch.Elasticsearch('localhost:9200')
```

[3]:
```
client.info()
```

[3]:
```
{'name': 'elastic_node01',
 'cluster_name': 'docker-cluster',
 'cluster_uuid': 'esd_btqvTTi7lyCc95hrwQ',
 'version': {'number': '7.9.2',
  'build_flavor': 'default',
  'build_type': 'docker',
  'build_hash': 'd34da0ea4a966c4e49417f2da2f244e3e97b4e6e',
  'build_date': '2020-09-23T00:45:33.626720Z',
  'build_snapshot': False,
  'lucene_version': '8.6.2',
  'minimum_wire_compatibility_version': '6.8.0',
  'minimum_index_compatibility_version': '6.0.0-beta1'},
 'tagline': 'You Know, for Search'}
```

2 Queries Simples

A sintaxe da query é exatamente a mesma que vimos para a interface REST.

[4]:
```
query = {
    "query": {
        "query_string": {
            "query": "enterprise Firefox"
        }
    }
}

results = client.search(index="kibana_sample_data_logs", body=query,␣
 ↪_source=["message"], size=20)
```

[5]:
```
print(f"Encontrado {results['hits']['total']['value']} resultados")
```

```
Encontrado 5741 resultados
```

[6]:
```
for hit in results['hits']['hits']:
    print(hit['_source']['message'])
```

```
177.111.217.54 - - [2018-07-22T03:37:04.863Z] "GET /enterprise_1 HTTP/1.1" 200
2492 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.1.4322)"
177.111.217.54 - - [2018-07-22T03:37:04.863Z] "GET /enterprise HTTP/1.1" 200
2492 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.1.4322)"
63.238.199.177 - - [2018-07-22T15:52:07.797Z] "GET /enterprise HTTP/1.1" 503 0
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
65.60.20.207 - - [2018-07-22T21:48:16.637Z] "GET /enterprise HTTP/1.1" 200 8909
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
100.47.209.39 - - [2018-07-22T15:29:12.789Z] "GET /enterprise HTTP/1.1" 200 6942
"-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
44.57.190.8 - - [2018-07-22T11:14:36.024Z] "GET /enterprise HTTP/1.1" 200 4064
"-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.50 Safari/534.24"
168.207.246.199 - - [2018-07-22T08:54:08.276Z] "GET /enterprise HTTP/1.1" 503 0
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
109.128.129.183 - - [2018-07-22T09:50:38.144Z] "GET /enterprise HTTP/1.1" 200
1603 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.1.4322)"
7.210.210.41 - - [2018-07-22T06:56:14.264Z] "GET /enterprise HTTP/1.1" 200 2053
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
184.50.180.99 - - [2018-07-22T10:08:30.405Z] "GET /enterprise HTTP/1.1" 200 4061
"-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.50 Safari/534.24"
88.13.107.162 - - [2018-07-22T11:08:50.594Z] "GET /enterprise HTTP/1.1" 200 9717
```

```
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
50.101.245.255 - - [2018-07-23T07:46:49.580Z] "GET /enterprise HTTP/1.1" 404
8655 "-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.50 Safari/534.24"
60.46.123.55 - - [2018-07-23T12:21:39.634Z] "GET /enterprise HTTP/1.1" 200 4695
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
33.16.170.252 - - [2018-07-23T12:48:03.093Z] "GET /enterprise HTTP/1.1" 200 4961
"-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.50 Safari/534.24"
155.21.88.247 - - [2018-07-23T14:32:58.561Z] "GET /enterprise HTTP/1.1" 200 7270
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
3.153.166.21 - - [2018-07-23T17:43:02.801Z] "GET /enterprise HTTP/1.1" 200 6898
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
120.188.47.64 - - [2018-07-23T11:12:23.938Z] "GET /enterprise HTTP/1.1" 200 8190
"-" "Mozilla/5.0 (X11; Linux x86_64; rv:6.0a1) Gecko/20110421 Firefox/6.0a1"
50.184.59.162 - - [2018-07-23T11:53:28.872Z] "GET /enterprise HTTP/1.1" 200 6833
"-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
111.51.93.126 - - [2018-07-23T14:33:28.643Z] "GET /enterprise HTTP/1.1" 200 1807
"-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.50 Safari/534.24"
92.17.158.57 - - [2018-07-23T18:27:17.763Z] "GET /enterprise HTTP/1.1" 404 7710
"-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)"
```

2.1 Agregaçoes Simples

Exemplo com Date Histogram

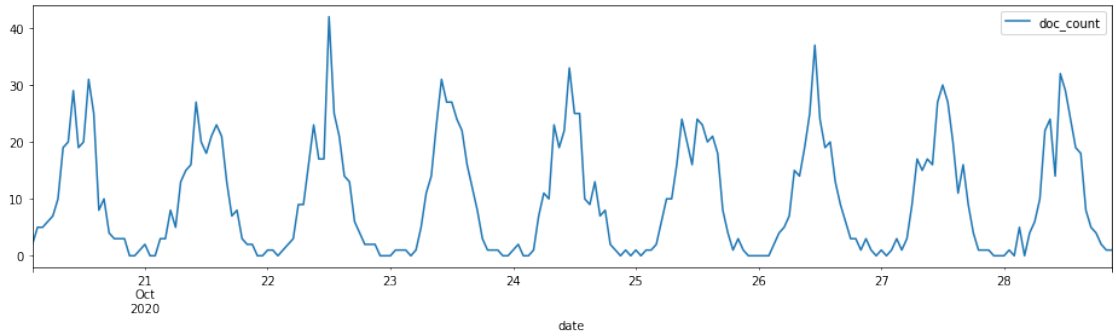
[7]:
```
query = {
  "aggregations": {
    "msgs_por_dia": {
      "date_histogram": {
        "field": "@timestamp",
        "calendar_interval": "1h"
      }
    }
  },
  "query": {
    "range": {
      "@timestamp":{"gte": "2020-10-20", "lte": "2020-10-28"}}
  }
}

results = client.search(index="kibana_sample_data_logs", body=query, size=0)
```

[8]:
```
buckets = results['aggregations']['msgs_por_dia']['buckets']
```

[9]:
```
df_buckets = pd.DataFrame.from_records(buckets)
df_buckets['date'] = pd.to_datetime(df_buckets['key_as_string'])
```

[10]:
```
df_buckets.plot(x='date', y='doc_count', figsize=(16, 4))
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64e6443d30>
```



2.2 Exemplo com agregação aninhada

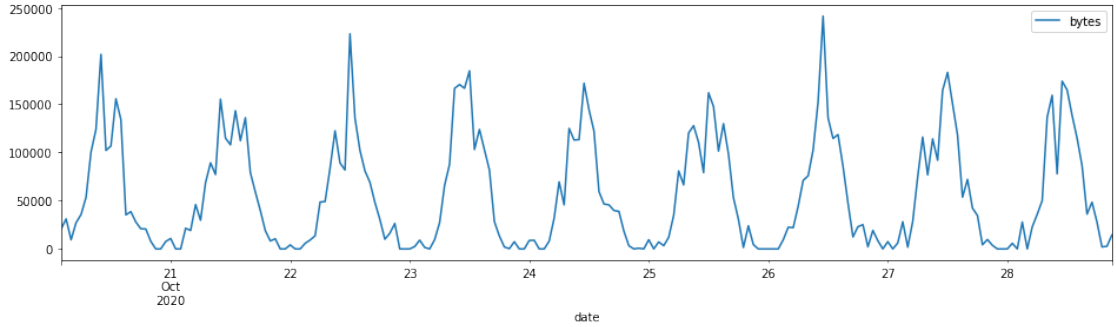
[11]:
```
query = {
  "aggregations": {
    "msgs_por_dia": {
      "date_histogram": {
        "field": "@timestamp",
        "calendar_interval": "1h",

      },
      "aggregations": {
        "bytes": {
            "sum": {"field": "bytes"}
        }
      }
    }
  },
  "query": {
    "range": {
      "@timestamp":{"gte": "2020-10-20", "lte": "2020-10-28"}}
  }
}

results = client.search(index="kibana_sample_data_logs", body=query, size=0)
```

[12]:
```
buckets = results['aggregations']['msgs_por_dia']['buckets']
```

[13]:
```
df_buckets = pd.DataFrame.from_records({"key_as_string": x['key_as_string'],␣
 ↪"bytes": x["bytes"]["value"]} for x in buckets)
df_buckets['date'] = pd.to_datetime(df_buckets['key_as_string'])
```

[14]:
```
df_buckets.plot(x='date', y='bytes', figsize=(16, 4))
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64e621bba8>
```



3 Criando um índice especificando o mapping

[15]:
```
# Apagar caso o indice exista
if client.indices.exists('20newsgroup_2'):
    client.indices.delete('20newsgroup_2')
```

[16]:
```
mapping = {
    "settings": {
        "number_of_shards": 3
    },
    "mappings": {
        "properties": {
            "newsgroup": {
                "type": "text",
                "fields": {
                    "raw": {
                        "type": "keyword"
                    }
                }
            },
            "message": {
                "type": "text",
                "fielddata": True,
                "fielddata_frequency_filter": {
                    "min": 0.01,
                    "min_segment_size": 10,
                }
            }
        }
    }
}
```

[17]:
```
client.indices.create('20newsgroup_2', body=mapping)
```

[17]:
```
{'acknowledged': True, 'shards_acknowledged': True, 'index': '20newsgroup_2'}
```

[18]:
```
with open('20_newsgroup.ndjson','r') as fin:
    for line in fin:
        document = json.loads(line)
        client.index(index='20newsgroup_2', body=document)
```

4 Agregação por termos no campo texto

[19]:
```
from nltk.corpus import stopwords
# Executar somente da primeira vez, para baixar a lista de stopwords
#nltk.download('stopwords')
```

[20]:
```
stopwords_en = stopwords.words('english')

stopwords_en += [
    'from', 'subject', 'lines', 'organization', 'nntp', 'posting', 'hosts',
    'writes', '[0-9]+','host', 'article', 'university', "i'm", "i've", 'would',
    'like', 'one', 'distribution', 'new', 'know', 'get', 'think', 'even', 'go',
    'say', 'many', 'time', 'want', 'much', 'us', 'people', 'good', 'could',
    'also', 'reply',
]
```

[21]:
```
query = {
  "aggregations": {
    "termos": {
      "terms": {
        "field": "message",
        "size": 100,
        "exclude": "(" + "|".join(stopwords_en) + ")",
      }
    }
  },
  "query": {"match_all": {}},
  "size": 0
}

results = client.search(index="20newsgroup_2", body=query)
```

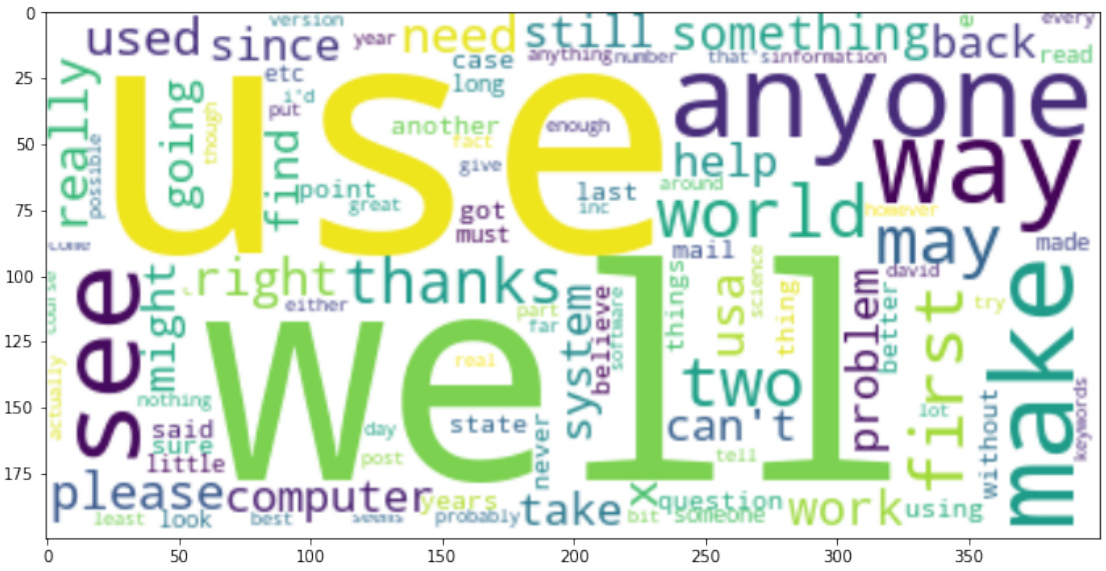
[22]:
```
buckets = results['aggregations']['termos']['buckets']
```

4.1 Plotando uma Wordcloud

[23]:
```
# Para instalar esse pacote, execute:
# pip install wordcloud
import wordcloud
```

[24]:
```
frequencies = {}
for bucket in buckets:
    frequencies[bucket['key']] = bucket['doc_count']

cloud = wordcloud.WordCloud(background_color="white")

cloud.generate_from_frequencies(frequencies)

fig, ax = plt.subplots(figsize=(16,6))
ax.imshow(cloud, interpolation='bilinear')
```

[24]:
```
<matplotlib.image.AxesImage at 0x7f64ae9152e8>
```



5 Wordcloud para rec.autos

[25]:
```
query = {
  "aggregations": {
    "termos": {
      "terms": {
        "field": "message",
        "size": 100,
        "exclude": "(" + "|".join(stopwords_en) + ")",
```

```
      }
    }
  },
  "query": {"term": {"newsgroup.raw": "rec.autos"}},
  "size": 0
}

results = client.search(index="20newsgroup_2", body=query)
buckets = results['aggregations']['termos']['buckets']
frequencies = {}
for bucket in buckets:
    frequencies[bucket['key']] = bucket['doc_count']

cloud = wordcloud.WordCloud(background_color="white")

cloud.generate_from_frequencies(frequencies)

fig, ax = plt.subplots(figsize=(16,6))
ax.imshow(cloud, interpolation='bilinear')
ax.axis('off')
_ = ax.set_title('Terms Aggregation para rec.autos')
```

5.1 Wordcloud com Significant Terms

O Significant Terms busca selecionar palavras que mais diferenciam o conjunto de documentos selecionados do *background*. Portanto, deve retornar termos que são mais significativos para o subconjunto de documentos selecionado pela query.

Veja na documentação do Significant Terms Aggregation as diferentes formas de selecionar a importância das palavras.

[26]:
```
query = {
  "aggregations": {
    "termos": {
      "significant_terms": {
        "field": "message",
        "size": 100,
        "exclude": "(" + "|".join(stopwords_en) + ")",
        "mutual_information": {
          "include_negatives": True
         }
      }
    }
  },
  "query": {"term": {"newsgroup.raw": "rec.autos"}},
  "size": 0
}

results = client.search(index="20newsgroup_2", body=query)
buckets = results['aggregations']['termos']['buckets']
frequencies = {}
for bucket in buckets:
    frequencies[bucket['key']] = bucket['doc_count']

cloud = wordcloud.WordCloud(background_color="white")

cloud.generate_from_frequencies(frequencies)

fig, ax = plt.subplots(figsize=(16,6))
ax.imshow(cloud, interpolation='bilinear')
ax.axis('off')
_ = ax.set_title('Significant Terms para rec.autos')
```

5.2 Significant Terms aggregation, mas para outro grupo

[27]:
```
query = {
  "aggregations": {
    "termos": {
      "significant_terms": {
        "field": "message",
        "size": 100,
        "exclude": "(" + "|".join(stopwords_en) + ")",
          "mutual_information": {
          "include_negatives": True
         }
      }
    }
  },
  "query": {"term": {"newsgroup.raw": "talk.politics.misc"}},
  "size": 0
}

results = client.search(index="20newsgroup_2", body=query)
buckets = results['aggregations']['termos']['buckets']
frequencies = {}
for bucket in buckets:
    frequencies[bucket['key']] = bucket['doc_count']

cloud = wordcloud.WordCloud(background_color="white")
```

```
cloud.generate_from_frequencies(frequencies)

fig, ax = plt.subplots(figsize=(16,6))
ax.imshow(cloud, interpolation='bilinear')
ax.axis('off')
_ = ax.set_title('Significant Terms para talk.politics.misc')
```