Pythonist (https://soumilshah1995.blogspot.com/)

Saturday, May 16, 2020

KNN Machine learning Algorithm on ElasticSearch

KNN Machine learning Algorithm on ElasticSearch

Step 1

Import the library

In [1]:

```
try:
    import elasticsearch
    from elasticsearch import Elasticsearch

    import pandas as pd
    import json
    from ast import literal_eval
    from tqdm import tqdm
    import datetime
    import os
    import sys
    import numpy as np

    from elasticsearch import helpers
    print("Loaded  .. . . . . . . .")
except Exception as E:
    print("Some Modules are Missing {} ".format(e))
```

```
Loaded  .. . . . . . . .
```

In [21]:

```
ENDPOINT = "http://localhost:9200"
```

In [29]:

```
es = Elasticsearch(timeout=600,hosts=ENDPOINT)
es.ping()
```

Out[29]:

```
True
```

Step 2:

- Preprocessing

Reading the Dataset

In [4]:

```
os.listdir()
```

Out[4]:

```
['.ipynb_checkpoints', 'netflix_titles.csv', 'Untitled.ipynb']
```

In [5]:

```
df= pd.read_csv("netflix_titles.csv")
```

In [7]:

```
df.head(1)
```

Out[7]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	descrip
0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	September 9, 2019	2019	TV-PG	90 min	Children & Family Movies, Comedies	Before planning awesome wedding for g

In [8]:

```
titles = df["title"].to_list()
```

In [9]:

```
len(titles)
```

Out[9]:

```
6234
```

Step 3:

Convert the Title into Vector using Google Pre trained Machine Learning Model

In [10]:

```
import tensorflow as tf
import tensorflow_hub as hub

module_url = "https://tfhub.dev/google/nnlm-en-dim128/2"
embed = hub.KerasLayer(module_url)

vector = []

for c, title in enumerate(titles):
    x = tf.constant([title])
    embeddings = embed(x)
    x = np.asarray(embeddings)
    x = x[0].tolist()
    vector.append(x)
```

In [11]:

```
len(vector)
```

Out[11]:

```
6234
```

In [12]:

```
vector[0]
```

Out[12]:

```
[0.08594007790088654,
 -0.09169773757457733,
 -0.08221833407878876,
 0.1603367030620575,
 0.05244443565607071,
 0.11267174780368805,
 -0.083820849657058 72,
 0.09882047772407532,
 0.021728241816163063,
 -0.18144601583480835,
 0.0012927550124004483,
 0.030685214325785637,
 -0.04533662274479866,
 -0.07281118631362915,
 -0.11955679953098297,
 0.017013853415846825,
 0.033623743802309036,
 -0.009736376814544201,
 0.033763282001018524,
 0.1921098679304123,
 0.00620125001296401,
 0.015555041842162609,
 0.06574436277151108,
 0.11323074996471405,
 -0.10774067789316177,
 0.1693897843360901,
 -0.13922490179538727,
 -0.10309454798698425,
 -5.2244857215555385e-05,
 0.023089049383997917,
 0.04559335112571716,
 -0.10510903596878052,
 -0.1005614772439003,
 -0.07881765812635422,
 0.025743374601006508,
 -0.05974612385034561,
 -0.1747055947780609,
 -0.05892287939786911,
 -0.06596986949443817,
 -0.09151236712932587,
 0.03593139722943306,
 -0.07345644384622574,
 -0.018012331798672676,
 0.036221787333488464,
 0.07314501702785492,
 -0.06195896118879318,
 -0.0023348417598754168,
 -0.1982719600200653,
 -0.3291093707084656,
 0.006821473129093647,
 0.1486814171075821,
 0.2550199031829834,
 0.1663597822189331,
 0.15605349838733673,
 0.12756910920143127,
 -0.057475071400403976,
 0.14456160366535187,
 -0.05416375771164894,
 0.06393317133188248,
 -0.08582285046577454,
 0.019529936835169792,
 0.030426720157265663,
 -0.13159017264842987,
 -0.01176383811980486,
 -0.05212199687957764,
 -0.007775180973112583,
 0.0005310662090778351,
 0.03532465547323227,
 0.14036867022514343,
 -0.04217003658413887,
 -0.0504852756857872,
 0.08859632164239883,
 0.02489238791167736,
 0.036609407514333725,
 0.012656561098992825,
 -0.031059175729751587,
 0.13535012304782867,
 -0.07467728853225708,
 -0.00639297952875495,
 -0.007216154597699642,
 0.10756982862949371,
 -0.03459356725215912,
 0.05434964969754219,
 0.10563021898269653,
 -0.023835688829421997,
 -0.1384897232055664,
 -0.10662095248699188,
 -0.11560706794261932,
 -0.018126854673027992,
 -0.11542601138353348,
```

```
 0.05233073979616165,
 -0.08457083255052567,
 0.04891547933220863,
 0.048610806465148926,
 -0.0861951932311058,
 -0.1646905094385147,
 0.05879170447587967,
 -0.09346245974302292,
 0.21104931831359863,
 0.07167480885982513,
 0.09941790252923965,
 -0.04874766618013382,
 -0.11821635812520981,
 -0.11691499501466751,
 -0.04042290896177292,
 -0.035517025738954544,
 0.006470585707575083,
 0.07046835869550705,
 0.032006461173295975,
 -0.017604319378733635,
 0.1958240568637848,
 0.01993837021291256,
 -0.01663972996175289,
 0.11849723011255264,
 -0.10080186277627945,
 -0.009301570244133472,
 0.03264541178941727,
 -0.03453604504466057,
 -0.032728590071201324,
 -0.06038405001163483,
 -0.014748498797416687,
 -0.08714324235916138,
 0.0329294428229332,
 -0.04497246816754341,
 -0.0888349711894989,
 0.02692333422601223,
 0.18709281086921692,
 -0.002944737207144499]
```

Step 4:

Creating documents

In [13]:

```
requests = []
for i, doc in enumerate(titles):
    request = {}
    request["_op_type"] = "index"
    request["_index"] = "myml"
    request["_id"] = i
    request["title"] = doc
    request["title_vector"] = vector[i]
    requests.append(request)
```

In [14]:

```
requests[0]
```

Out[14]:


```
{'_op_type': 'index',
 '_index': 'myml',
 '_id': 0,
 'title': 'Norm of the North: King Sized Adventure',
 'title_vector': [0.08594007790088654,
  -0.09169773757457733,
  -0.08221833407878876,
  0.1603367030620575,
  0.05244443565607071,
  0.11267174780368805,
  -0.08382084965705872,
  0.09882047772407532,
  0.021728241816163063,
  -0.18144601583480835,
  0.0012927550124004483,
  0.030685214325785637,
  -0.04533662274479866,
  -0.07281118631362915,
  -0.11955679953098297,
  0.017013853415846825,
  0.033623743802309036,
  -0.009736376814544201,
  0.033763282001018524,
  0.1921098679304123,
  0.00620125001296401,
  0.015555041842162609,
  0.06574436277151108,
  0.11323074996471405,
  -0.10774067789316177,
  0.1693897843360901,
  -0.13922490179538727,
  -0.10309454798698425,
  -5.2244857215555385e-05,
  0.023089049383997917,
  0.04559335112571716,
  -0.10510903596878052,
  -0.1005614772439003,
  -0.07881765812635422,
  0.025743374601006508,
  -0.05974612385034561,
  -0.1747055947780609,
  -0.05892287939786911,
  -0.06596986949443817,
  -0.09151236712932587,
  0.03593139722943306,
  -0.07345644384622574,
  -0.018012331798672676,
  0.036221787333488464,
  0.07314501702785492,
  -0.06195896118879318,
  -0.0023348417598754168,
  -0.1982719600200653,
  -0.3291093707084656,
  0.006821473129093647,
  0.1486814171075821,
  0.2550199031829834,
  0.1663597822189331,
  0.15605349838733673,
  0.12756910920143127,
  -0.057475071400403976,
  0.14456160366535187,
  -0.05416375771164894,
  0.06393317133188248,
  -0.08582285046577454,
  0.019529936835169792,
  0.030426720157265663,
  -0.13159017264842987,
  -0.01176383811980486,
  -0.05212199687957764,
  -0.007775180973112583,
  0.0005310662090778351,
  0.03532465547323227,
  0.14036867022514343,
  -0.04217003658413887,
  -0.0504852756857872,
  0.08859632164239883,
  0.02489238791167736,
  0.036609407514333725,
  0.012656561098992825,
  -0.031059175729751587,
  0.13535012304782867,
  -0.07467728853225708,
  -0.00639297952875495,
  -0.007216154597699642,
  0.10756982862949371,
  -0.03459356725215912,
  0.05434964969754219,
  0.10563021898269653,
  -0.023835688829421997,
  -0.1384897232055664,
```

```
  -0.10662095248699188,
  -0.11560706794261932,
  -0.018126854673027992,
  -0.11542601138353348,
  0.05233073979616165,
  -0.0845708325505256 7,
  0.04891547933220863,
  0.048610806465148926,
  -0.0861951932311058,
  -0.1646905094385147,
  0.05879170447587967,
  -0.09346245974302292,
  0.21104931831359863,
  0.07167480885982513,
  0.09941790252923965,
  -0.04874766618013382,
  -0.11821635812520981,
  -0.11691499501466751,
  -0.04042290896177292,
  -0.035517025738954544,
  0.006470585707575083,
  0.07046835869550705,
  0.032006461173295975,
  -0.017604319378733635,
  0.1958240568637848,
  0.01993837021291256,
  -0.01663972996175289,
  0.11849723011255264,
  -0.10080186277627945,
  -0.009301570244133472,
  0.03264541178941727,
  -0.03453604504466057,
  -0.032728590071201324,
  -0.06038405001163483,
  -0.014748498797416687,
  -0.08714324235916138,
  0.0329294428229332,
  -0.04497246816754341,
  -0.0888349711894989,
  0.02692333422601223,
  0.18709281086921692,
  -0.002944737207144499]}
```

Define mappings

In [30]:

```
settings ={
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 1,
    "index.knn": True
  },
   "mappings": {
    "dynamic": "true",
    "_source": {
      "enabled": "true"
    },
    "properties": {
      "title": {
        "type": "text"
      },
      "title_vector": {
        "type": "knn_vector",
        "dimension": 128
      }
    }
  }
}
```

In [31]:

```
IndexName = 'myml'
my = es.indices.create(index=IndexName, ignore=[400,404], body=settings)
```

In [32]:

```
my
```

Out[32]:

```
{'acknowledged': True, 'shards_acknowledged': True, 'index': 'myml'}
```

Step 5:

load into elastic search

In [33]:

```
try:
    res = helpers.bulk(es, requests)
    print("Working")
except Exception as e:
    print(e)
```

```
Working
```

Testing KNN model

we are using cosine similarity to get result in ELK

In [36]:

```
title = input("Enter query: ")

x = tf.constant([title])
embeddings = embed(x)
x = np.asarray(embeddings)
x = x[0].tolist()


script_query = {
    "script_score": {
        "query": {"match_all": {}},
        "script": {
            "source": "cosineSimilarity(params.query_vector, doc['title_vector']) + 1.0",
            "params": {"query_vector": x}
        }
    }
}

script_query = {
    "knn": {
        "title_vector": {
            "vector": x,
            "k": 2
        }
    }
}

response = es.search(
    index="myml",
    body={
        "size": 10,
        "query": script_query,
        "_source": {"includes": ["title", "body"]}
    }
)

for hit in response["hits"]["hits"]:
    print("id: {}, score: {}".format(hit["_id"], hit["_score"]))
    print(hit["_source"])
    print()
```

```
Enter query: Swiss Army Man
id: 3241, score: 1.0
{'title': 'Swiss Army Man'}

id: 1349, score: 0.51294893
{'title': 'American Son'}

id: 6150, score: 0.48458296
{'title': 'Glitter Force'}

id: 4484, score: 0.48386106
{'title': 'A Family Man'}

id: 5785, score: 0.47989023
{'title': 'American Crime'}

id: 3784, score: 0.4735783
{'title': 'Star Men'}

id: 953, score: 0.46827134
{'title': 'Phantom Boy'}

id: 5621, score: 0.46712905
{'title': 'American Vandal'}

id: 5723, score: 0.46373478
{'title': 'Man Down'}

id: 4167, score: 0.46257648
{'title': 'Mercenary'}
```

at May 16, 2020 (2020-05-16T04:48:00-07:00) (https://soumilshah1995.blogspot.com/2020/05/knn-machine-learning-algorithm-on.html)

2 comments:

Riziq (https://www.blogger.com/profile/07679256568190925445) September 23, 2020 at 12:44 AM (https://soumilshah1995.blogspot.com/2020/05/knn-machine-learning showComment=1600847083329#c8362534923915409747)

Hi, when i try to execute "my = es.indices.create(index=IndexName, ignore=[400,404], body=settings)" why always show error

{'error': {'root_cause': [{'type': 'illegal_argument_exception',
'reason': 'unknown setting [index.knn] please check that any required plugins are installed, or check the breaking changes documentation for removed settings'}],
'type': 'illegal_argument_exception',
'reason': 'unknown setting [index.knn] please check that any required plugins are installed, or check the breaking changes documentation for removed settings'},

'status': 400}

Do you have solutions?

Reply

recyclage (https://www.blogger.com/profile/05702849945738149872) February 12, 2021 at 4:59 PM (https://soumilshah1995.blogspot.com/2020/05/knn-machine-learnir showComment=1613177946906#c1085004386485463375)

Please, How do I fix error message : RequestError: RequestError(400, 'parsing_exception', 'unknown query [knn]')

Reply

Enter your comment...

Comment as: hemanth22he

Publish Preview

(https://www.blogger.com/comment-iframe.g?blogID=2397361725226431430&postID=1481237809796196734&blogspotRpcToken=87581)

Newer Post (https://soumilshah1995.blogspot.com/2020/05/parallelizing-word2vec-in-python-with.html) Home (https://soumilshah1995.blogs
Older Post (https://soumilshah1995.blogspot.com/2020/05/name-entity-recognition-

Subscribe to: Post Comments (Atom) (https://soumilshah1995.blogspot.com/feeds/1481237809796196734/comments/default)

Power of Semantics Search combined with Elastic Search | ML on ELK (https://soumilshah1995.blogspot.com/2022/01/power-of-semantics-search-combined-with.html)

master Power of Semantics Search combined with Elastic Search | ML on ELK ¶ Soumil ...

Project: Data Analysis and Visualizations and Predicting Future Energy Consumption using LSTM Predicting Values 2 month Later Accurately RNN (https://soumilshah1995.blogspo data-analysis-and.html)
Energy Hourly Energy Consumption ¶ Step 1: ¶ Import Library ¶ I...

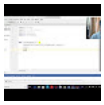
(https://soumilshah1995.blogspot.com/2019/04/server-and-client-send-actual-sensor.html)

Server and Client Send Actual Sensor Data over Network using Raspberry (https://soumilshah1995.blogspot.com/2019/04/server-and-client-send-actua
Lab 3 Server and Client Send Actual Sensor Data over Network using Rasp

(https://soumilshah1995.blogspot.com/2019/05/smart-proxy-library-to-get-random-proxy.html)

Smart Proxy library to get random proxy using Python [Hide your Identity (https://soumilshah1995.blogspot.com/2019/05/smart-proxy-library-to-ge
Smart Library that's fetch Random Proxy using Python Smart Proxy libra

Name Entity Recognition on PDF Resume using NLP and spacy python (https://soumilshah1995.blogspot.com/2020/05/name-entity-recognition-on-pdf-resume.html)
NamedEntity Name Entity Recognition on PDF Resume using NLP and spacy ¶ In [22...



(https://soumilshah1995.blogspot.com/2019/04/upload-any-sensor-data-to-thingspeak.html)

Upload any Sensor data to ThingSpeak using Raspberry/Arduino Python (Examples (https://soumilshah1995.blogspot.com/2019/04/upload-any-sen thingspeak.html)
Lab 4 (ThingSpeak) Getting started with Open Source Cloud Server Uploa

Getting started with Elastic Search and Python (https://soumilshah1995.blogspot.com/2020/01/getting-started-with-elastic-search-and.html)
Getting started with Elastic Search Getting started with Elastic Search and Python ¶...

4 Ways to do Pagination or scrolling in Elastic Search Tutorials (https://soumilshah1995.blogspot.com/2020/06/elk-pre-margin-0px-border-none-padding.html)
ELK Elastic Search Tutorials ¶ 4 Ways to do Pagination or scrolling in Elastic Searc...

Using BERT with Scikit Learn to do Text classification¶ (https://soumilshah1995.blogspot.com/2021/04/using-bert-with-scikit-learn-to-do-text.html)
BERT Using BERT with Scikit Learn to do Text classification ¶ Soumil Nitin Shah ¶ Ba...

KNN Machine learning Algorithm on ElasticSearch (https://soumilshah1995.blogspot.com/2020/05/knn-machine-learning-algorithm-on.html)
Untitled KNN Machine learning Algorithm on ElasticSearch ¶ Step 1 ¶ Import th...

Entity Recognition Extract information from Job posting using Spacy Machine learning Model (https://soumilshah1995.blogspot.com/2021/04/entity-recognition-extract-information.ht
Untitled Entity Recognition Extract information from Job posting ¶ Soumil Nitin Sha...

Followers

Followers (17)

Follow

Contact Form

Name

Email *

Message *

Send