Hotel Indexer:

Auther: Esraa Madi

Overview

When you are running a website that provides lots of dynamic content; be it an e-commerce website or a blog. You need not only provide a robust search engine for your web app but you need also provide native auto-complete features in your app. that is what ElasticSearch could provide for our website.

Elasticsearch is a real-time distributed search and analytics engine. It allows you to explore your data at a speed and at a scale never before possible. It is used for full text search, structured search, analytics and all three in combination. Elastic search is an open source search engine built on top of Apache Lucecne, a full text search engine library.

How Hotel Indexer Service Works?

- When you hit this endpoint it should use ElasticSearch to index all data found in dataset for each hotel
- Each hotel MUST have ONLY one document with all its data.

Datasets

The dataset used in this service is the raw data taken from this Kaggle dataset (https://www.kaggle.com/datafiniti/hotel-reviews#7282_1.csv) **plus** the data obtained from Watson lib dataset ()

This dataset is a list of about 1,000 hotels and 30,000 reviews. The dataset includes hotel location, name, rating, review data, title, username, and more.

For more information about dataset columns, you can check this data dictionary (https://developer.datafiniti.co/docs/business-data-schema)

Installing and running Elasticsearch:

To get started with Elasticsearch, I found this link (https://www.elastic.co/downloads/elasticsearch) is very helpful.

In [ ]:

```
#!pip install elasticsearch
```

Let's start :)

In [16]:

```
# import libraries
import pandas as pd
import pickle
```

In [4]:

```
# Read the dataset
hotel_data = pd.read_csv('../data/processed_data/hotel_review_text_tone.csv', in
dex_col=0)
```

In [5]:

```
# check data header
hotel_data.head(2)
```

Out[5]:

	address	categories	city	country	latitude	longitude	name	postalCode	
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	(
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	(

2 rows × 34 columns

In [6]:

```
# check missing values
hotel_data.isnull().sum()
```

Out[6]:

```
address                        0
categories                     0
city                           0
country                        0
latitude                      82
longitude                     82
name                           0
postalCode                    55
province                       0
reviews_date                 232
reviews_dateAdded              0
reviews_doRecommend        33604
reviews_id                 33604
reviews_rating               491
reviews_text                   0
reviews_title               1153
reviews_userCity           18911
reviews_username              41
reviews_userProvince       17594
is_hotel                       0
review_anger                   0
review_fear                    0
review_joy                     0
review_sadness                 0
review_analytical              0
review_confident               0
review_tentative               0
review_title_anger             0
review_title_fear              0
review_title_joy               0
review_title_sadness           0
review_title_analytical        0
review_title_confident         0
review_title_tentative         0
dtype: int64
```

Elasticsearch cannot handle the `None` type, so we need to fill all missing values.

In [7]:

```
# Fill missing columns with default values depending on columns type
hotel_data.latitude.fillna(0, inplace=True)
hotel_data.longitude.fillna(0, inplace=True)
hotel_data.postalCode.fillna(' ', inplace=True)
hotel_data.reviews_date.fillna('1111-11-11T00:00:00Z', inplace=True)
hotel_data.reviews_doRecommend.fillna(0, inplace=True)
hotel_data.reviews_id.fillna(0, inplace=True)
hotel_data.reviews_rating.fillna(0, inplace=True)
hotel_data.reviews_title.fillna(' ', inplace=True)
hotel_data.reviews_userCity.fillna(' ', inplace=True)
hotel_data.reviews_username.fillna(' ', inplace=True)
hotel_data.reviews_userProvince.fillna(' ', inplace=True)
```

> **Elasticsearch** is document oriented, meaning that it stores entire object or documents. It not only stores them, but also indexes the content of each document in order to make them searchable.

So, let's bulid a doc for each hotel

In [8]:

```
def create_hotel_doc(df):
    '''
    Function to create a (doc) one dictionary for each hotel, contains all its d
ata in the dataset
    *Args:
        df: dataframe, given dataframe for a hotel
    *Return:
        hotel_dict: dict, docs contains hotel data
    '''

    # store hotel informations ( address, type , long, ...)
    hotel_dict = df.iloc[0,:9].to_dict()

    # store hotel reviews as a list inside the above dict and this list contains
a dictionary for each review
    hotel_dict['reviews'] = df.iloc[:,9:].to_dict(orient='records')

    return hotel_dict
```

In [9]:

```
# list to hold hotels docs
hotels_docs = []

# loop over hotels
for hotel_name in hotel_data.name.unique():

    # create and append dict for a hotel
    dict_ = create_hotel_doc(hotel_data[hotel_data.name == hotel_name])
    hotels_docs.append(dict_)
```

In [10]:

```
print('We have {} docs in our list, each doc is a hotel data'.format(len(hotels_
docs)))
```

```
We have 725 docs in our list, each doc is a hotel data
```

In [15]:

```
# check how hotel doc looks like
hotels_docs[720]
```

Out[15]:


```
{'address': 'N4449 Us Highway 45',
 'categories': 'Hotels,Lodging,Bed & Breakfast & Inns,Motels',
 'city': 'Tigerton',
 'country': 'US',
 'latitude': 44.750305,
 'longitude': -89.06617,
 'name': 'Rock A Bye Inn',
 'postalCode': '54486',
 'province': 'Morris',
 'reviews': [{'reviews_date': '1111-11-11T00:00:00Z',
   'reviews_dateAdded': '2016-04-03T12:14:00Z',
   'reviews_doRecommend': 0.0,
   'reviews_id': 0.0,
   'reviews_rating': 0.0,
   'reviews_text': 'to share your opinion of this businesswith YP vi
sitors across the United Statesand in your neighborhood',
   'reviews_title': ' ',
   'reviews_userCity': ' ',
   'reviews_username': 'write a review',
   'reviews_userProvince': ' ',
   'is_hotel': 1,
   'review_anger': 0.0,
   'review_fear': 0.0,
   'review_joy': 0.0,
   'review_sadness': 0.0,
   'review_analytical': 0.0,
   'review_confident': 0.0,
   'review_tentative': 0.0,
   'review_title_anger': 0.0,
   'review_title_fear': 0.0,
   'review_title_joy': 0.0,
   'review_title_sadness': 0.0,
   'review_title_analytical': 0.0,
   'review_title_confident': 0.0,
   'review_title_tentative': 0.0}]}
```


As you can see above, we have created a dictionary for each hotel. For columns have fixed information about the hotel like address, categories, city and so on, each column became a key with a value in our dictionary. And for columns have reviews information like review text, review title, etc, we have a key called `reviews` and its value is a list of dictionaries, each dictionary has information about one user review

In [50]:

```
# save this list in a file to use it later on the flask
with open('../data/processed_data/hotels_docs', 'wb') as fp:
    pickle.dump(hotels_docs, fp)
```

Now, let's store these documents using Elasticsearch

In [19]:

```
# Import Elasticsearch libraries
from elasticsearch import Elasticsearch
```

In [20]:

```
# Connect to the elastic cluster
es=Elasticsearch([{'host':'localhost','port':9200}])
es
```

Out[20]:

```
<Elasticsearch([{'host': 'localhost', 'port': 9200}])>
```

Elasticsearch Index:

Is like a database in traditional database. It is the place to store related documents.

> To store or retrive any document we would need three pieces of informantion:
>
> - Index — Database
> - Datatype — Type of the document
> - Id — Id of the document

In [21]:

```
# store hotels docs in index called 'hotelreview' and give each doc an Id
id_ = 1
for doc in hotels_docs:
    res = es.index(index='hotelreview',doc_type='hotel',id=id_,body=doc)
    id_ += 1
```

Test retrieving one doc by one of its features

Try to find a hotel by looking for `Highway` word in the address

In [35]:

```
result = es.search(index="hotelreview", body={"query": {"match": {'address':'Hig
hway'}}})['hits']['hits']
```

In [47]:

```
print('The search query returns {} hotels docs'.format(len(result)))
```

```
The search query returns 10 hotels docs
```

In [46]:

```
for ind, hotel_doc in enumerate(result):
    print(ind+1, '- Hotel name:',hotel_doc['_source']['name'])
    print('\t Hotel Address:',hotel_doc['_source']['address'])
```

```
1 - Hotel name: Econo Lodge St Robert
         Hotel Address: 309 Highway Z
2 - Hotel name: Motor Inns of America
         Hotel Address: 4740 Highway 68
3 - Hotel name: Boiling Springs Resort Canoe Rental
         Hotel Address: 15750 Highway Bb
4 - Hotel name: Intercontinental Hotels Resorts
         Hotel Address: 17300 Highway 99
5 - Hotel name: Old Wheeler Hotel
         Hotel Address: 495 Highway 101
6 - Hotel name: Arrowhead Motel and Rv Park
         Hotel Address: 616 Highway 70
7 - Hotel name: Mineola Motel
         Hotel Address: 1505 Highway 17
8 - Hotel name: Monarch Inn
         Hotel Address: 5059 Highway 140
9 - Hotel name: Quality Inn Gulfport
         Hotel Address: 9435 Highway 49
10 - Hotel name: Sportsmen Motel
         Hotel Address: 2909 E Highway 101
```