Assignment 5: Learning Elasticsearch

Description:

Supported by the 2018_movie corpus, an information retrieval system is built with elasticsearch.

Dependencies:

MacOS High Sierra version 10.13.2
Python 3.6
Flask==1.0.2
Elasticsearch 6.7
Elasticsearch-dsl 6.7
Java 8

Run Instructions:

Activate elasticsearch:

```
bin/elasticsearch
```

Execute the following command to generate an Index called sample_film_index:

```
python3  index.py
```

Execute the following command to run the information retrieval system:

```
python3  query.py
```

Click on http://127.0.0.1:5000/ to run in a browser

Modules and methods:

index.py

Movie

A class sets a data type to each field in our json file.

Field analyzers:

my_analyzer: this is used in Title and Text field. It includes functionalities of case-folding, stopword filter and porter-stemmer.
standard analyzer: this is used in Starring, Director, Location, Categories, Country and Language field.
keyword analyzer: this is used in Time field and Runtime field. Because we need no preprocessing to be done in this field.

buildIndex:

creates a new film index, deleting any existing index of the same name.It loads a json file containing the movie corpus and does bulk loading using a generator function.

query.py

Customized queries for each field:

```
q = Q('multi_match', query=text_query, type='cross_fields',
fields=['title^2', 'text'], operator='and')
```

This is applied to title field and text field, the title field is boosted based on the assumptions that people the phrase is more likely to be in the title of the film than in the free text.

```
q = Q('fuzzy', starring={'value': star_query, 'transpositions':
True})
```

This is applied to the starring and director field. Because people are very likely to spell the people name wrong, therefore we use the transporsitions feature of the fuzzy query.

```
q = Q('match', categories={"query": cat_query,
"cutoff_frequency": 0.02})
```

This is applied to the categories field. The vocabulary in the categories field are very fixed. A lot of high frequency terms appear in it. Therefore cutoff_frequency is used to dynamically get rid of the high frequency terms.

```
q = Q('match', location=locat_query)
```

This is applied to the rest of the fields to get exactly the user input and put no additional weights.

Files in the folder:

2018_movies.json: the movie corpus that supported the boolean IR system

search_helper.py

index.py

query.py

templates/page_query.html, templates/page_SERP.html, pager_targetArticle.html, files used to create the web page