Artificial Intelligence & Cloud

Project Assignment - ElasticSearch

Main Goals

This project consists in the analysis of the powerful searching capabilities of ElasticSearch and the creation of an ElasticSearch-powered search engine for a movie's database.

Data specifications

The data collection proposed for the project is related to a collection of movies' information. It can be downloaded from the "Portale della Didattica" (imdb.json).
The collection includes information about movies and their associated reviews on IMDB and Rotten Tomatoes.

Data Indexing

The first step of the project consists in parsing and indexing relevant information from the original JSON data collection.

Write and execute a python script that parse and index (on specific document fields) the following information from the original data collection:

- Movie's title (`title`)
- Movie's plot (`plot`)
- Movie's publication year (`year`)
- Movie's full plot (`fullplot`)
- IMDB average rating (`imdb.rating`)
- Rotten Tomatoes average rating (`tomatoes.viewer.raing`)
- List of directors (`directors`)
- List of countries (`countries`)
- List of genres (`genres`)

If any of these fields do not exist for a document in the original collection, initialize it as
- empty ("" for strings),
- 0 value (for numeric fields) or,
- an empty list (`[]` for lists).

Interest requests

Using the python `elasticsearch` library, solve the following requests:

1. Get the list of the first 10 movies that have been published between 1939 and 1945 and best matches the word "war" in the plot.
2. Get the list of the first 20 movies that have an average IMDB rating higher than 6.0 and best matches "Iron Man" in the full plot.
3. Search for the movies that best match the text query "matrix" in the title. Boost the results by multiplying the standard score with the IMDB rating score. Repeat the same query considering the Rotten Tomatoes score instead of IMDB. Does the order of the results change?

NB: To apply a score modifier you can use the native `script_score` function. To avoid runtime errors, you should check in your short script if the doc elements contain the desired key (e.g., `imdb_score`).
E.g. `(doc.containsKey('field') ? doc['field'].value : 1)`

Movies search engine

Create a flask-based application that is capable of performing full-text search over the provided data collection. The application should include:

- An `highlighter` that shows the first match of the full-text search in bold. The highlighter could be implemented both in `plot` or `fullplot` fields, the choice is left to your preference.
- A score booster that is able to increase the score using IMDB ratings (or Rotten Tomatoes ratings, the choice is left to your preference).
- A results.html page that presents the title and the first match of the highlighter for each of the top-20 documents after the query search.

Project assignment

Report format

Write a report of including the following information:

1. **Project overview**
2. **Interest queries**: explain the three interest queries and the main operators involved in this phase.
3. **Movies search engine**: describe the designed flask application. Please provide specific insights on the impact of boosting standard scores by using the IMDB/Rotten Tomatoes rating values.
4. **Scoring function analysis**: analyze, from a critical point of view, the standard scoring function used by ElasticSearch (TF-IDF). What are its strengths and weaknesses in a real scenario?
5. **Additional search request:** write and solve one additional novel request, defined by you, and describe when and how it could be helpful in a real use-case.

Constraints on the report

The report must comply with the following constraints:

- The report must be generated using the standard IEEE conference template, available in LATEX and Word format.
  You are strongly encouraged to use the LaTeX version (you can also use Overleaf).

- The report must follow the division in the listed sections.

- The report must be, at most, 4 pages long.

Submission

You should submit a single file (zip format) to the "Portale della Didattica" under the Homework section ("Elaborati"). The compressed folder must contain:

- The report in PDF format.
- The script used in the index data in ElasticSearch (ingest_data.py)
- The body of the interest requests (requests.txt).
- The complete code of the flask application (inside a folder named `movies_es`)

The file must be named "Project_3_ES.zip".

Deadline: April 7th, 2021, end of the day (23:59 CET)

In case of doubts or problems, please contact Moreno La Quatra (moreno.laquatra@polito.it)