Instructor: Jian Pei $<jpei@sfu.ca>$
TA: Weiyuan Wu $<youngw@sfu.ca>$
Nguyen Cao $<nguyen\_cao@sfu.ca>$

CMPT456: Assignment 4
Elasticsearch

2019/11/01

Due: November 17, 2019 (Sunday) at 11:59pm

This assignment should be done individually

Important Note

The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the TAs if you are having difficulties with this assignment.

DO NOT

1. Give/receive code or proofs to/from other students
2. Use Google to find solutions for assignment

DO

1. Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
2. Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

Getting Started

Elasticsearch is a distributed search and analytics engine and provides real-time search and analytics for all types of data. In this assignment you are asked to conduct text filtering, searching and ranking using Elasticsearch.

Before you start your project, it is suggested you to take a tour to get some basic idea about Elasticsearch.

Task Requirements

We will use Python (3.6.8) as well as the latest Elasticsearch (7.4.0) for this assignment. For easy installation and dependency control, same as A2, you will use Docker to finish all the tasks.

We are going to use a **customized** Wiki Small data (6043 documents) from our textbook. It is already included in the docker image at `/workdir/wiki-small.tar.gz`.

All the commands involved in this document assume you run them on *nix machines. For Windows machines, replace `$PWD` with `%cd%`.

Note: All the searching and ranking tasks should be done using Elasticsearch queries. Loading all the text locally and searching using Python is not allowed!

The Tasks

The code is already on our Gitlab. It contains four python files 'assignment3.py', 'q1.py', 'q2.py' and 'q3.py', as well as a stopword list 'stopwords.txt' and the wiki data 'wiki-small.tar.gz'. This assignment requires you to complete the functions in the python files to finish a search task: finding a place to take a trip. In detail, you are asked to first configure the Elasticsearch index and load the data. After that, you need to conduct some text filtering using Elasticsearch API. Then, you need to rank the result based on a criterion.

We already built the docker image for you, containing Elasticsearch, the wiki data, as well as the python dependencies `PyQuery` and `Elasticsearch`. It is highly recommended to use the prebuilt image since it already contains all the necessary dependencies. The prebuilt image can be found here. In the case you want to build your own docker image, the build script can be found here. We **do not** provide support on custom-built images.

After you finished the assignment, please make sure the following commands are runnable under the directory containing folder 'src' (The answers will be scored based on the output of these commands as well as the code you write!):

```
docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q1.py
docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q2.py
docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q3.py
```

For easy coding, you can also run

```
docker run -it --rm -v $PWD/src:/workdir/src -p9200:9200 -p9300:9300 wooya/cmpt456a4 standby
```

This command will start an Elasticsearch instance so that you can directly run python scripts on your machine.

Demo

To check everything is set properly, clone the codebase into a folder. Into that folder, you run

```
docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 demo.py
```

If everything works well, you will see:

```
ES started
Hello world
```

In your terminal.

Assignment Description

Q1: Load the Data (30 pts)

In the first part of this assignment, you are going to create an index in the Elasticsearch with some configurations. After that, you will parse the wiki data and load them into the Elasticsearch.

1. Complete the function `assignment4.create_wikipedia_index`. This function creates an index called 'wikipedia', with an analyzer called `my_analyzer`.

   `my_analyzer` will use `lowercase` and `my_stops` as the filter, in which the latter is a stopword filter that loads the provided stopword list `stopwords.txt`. `my_analyzer` should also use the `standard` tokenizer.(15 pts)

   Hint:

   (a) Helpful links about Elasticsearch Mapping and Analyzers, Customized analyzer, Create Index API.

   (b) You need to mount the directory from your machine into the docker container. ref

   (c) Elasticsearch has security policies to prevent you from loading the stopword list from anywhere. Be sure to put the stopword list under `/usr/share/elasticsearch/config/` in the container.

2. Complete the function `assignment3.parse_html`. The raw HTML file contains many irrelavent HTML tags like `<div>` and `<body>`. It is easier to remove them before loading texts into Elasticsearch. Please write a function that parses the HTML text and extracts the text content from `<title>` and `<body>`. You are highly suggested to use `PyQuery` to parse the HTML and it is already installed inside the docker image. (5 pts)

3. Complete the function `assignment3.load_data`. This funciton will load the parsed HTML files (using the function `assignment3.parse_html`) into Elasticsearch under index 'wikipedia' from the tarball `wiki-small.tar.gz` (located at the container path `/workdir/wiki-small.tar.gz`). Each document should be a JSON with 2 fields: 'title' and 'body'.

   ```
   {
     'title': '...',
     'body': '...',
   }
   ```

   (5 pts)

4. Complete the function `q1.count_documents`. This function will query the Elasticsearch and return the number of documents contained in the 'wikipedia' index. (5 pts)

5. Make sure to run `docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q1.py` to check your answer is runnable!

Q2: Issue a Simple Query (25 pts)

In this question, you are asked to do some text filtering on the documents.

1. Issue a query to Elasticsearch on the 'wikipedia' index. The returned documents should contain the word "lake" or "tour". Complete the function 'q2.filter'. (15 pts)

2. Some of the documents contain the sentence "Please improve this article if you can.". These documents are undesirable in our searching result. Please modify the query in Q2.1 so that the documents containing the sentence "Please improve this article if you can." are removed from the searching result. Complete the function `q2.search_without_improvement`. (10 pts)

3. Make sure to run `docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q2.py` to check your answer is runnable!

Q3: Rank the Result (45 pts)

The search result from Q2 is not ranked yet. You can see lots of not quite related documents at the top of the search result. In this question, you need to rank these search results.

1. Based on the query in Q2.2, please rank the result by their location: the documents whose body contains the word “BC”, “WA” or “AB” should be ranked before other documents. Moreover, documents containing ”BC” should be most top-ranked. Unfortunately, Elasticsearch doesn’t recognize the abbreviations “BC”, “AB”,“WA”. Please configure the “wikipedia” index so that it knows “BC” = “British Columbia”, “AB” = “Alberta” and “WA” = “Washington”. Complete the function `q3.add_synonyms_to_index`. Note: The synonym filter should be named as `my_synonyms`. (20 pts)
2. Issue a search again that ranks the documents based on the query in Q2 as described in Q3.1. Complete the function `q3.search_and_rank`. (15 pts)
3. How does the rank work? Please write down how your codes in Q3.1 and Q3.2 change the order of the returned documents in `q3.how_does_rank_work` (10 pts)
4. Make sure to run `docker run -it --rm -v $PWD/src:/workdir/src wooya/cmpt456a4 q3.py` to check your answer is runnable!

Submit your Assignment

The assignment must be submitted online at https://coursys.sfu.ca. You need to submit the following files:

1. assignment4.py
2. q1.py
3. q2.py
4. q3.py