Final Report

Zongxian Feng

Content

The Problem

Unlike search engines like Microsoft Edge and Google Search which can be continuously updated, the academic search engine has a long-standing problem that publications can hardly change after they were published. In this case, an old publication with outdated keywords would be less likely to be searched by the user. Besides that, the popularity and the authority of a publication mainly rely on the citation number, which might increase the Matthew Effect.

My job is to develop an academic search engine demo. I downloaded the dataset from Aminer and Microsoft Academic Graph(MAG) and built the engine using ELK stack on Azure Virtual Machine.

My Approaches

1. Download datasets

We survey 7 popular bibliographic databases and decide to use Open Academic Graph(OAG) which is generated by linking two large academic graphs: Microsoft Academic Graph and Aminer. We use the snapshot of MAG in July 2020 and the snapshot of Aminer in October 2020 since both graphs are evolving.

I created a virtual machine (es-w1) on Azure. The configuration of the virtual machine is listed below:

Operating System	Windows (Windows Server 2019 Datacenter)
Size	Standard D8s v3 (8 vcpus, 32 GiB memory)
Location	East US
Public IP address	52.186.78.60
Private IP address	10.2.0.4
Username	zfeng13
Password	Bjky_13622361190

I connect the virtual machine using a remote desktop. I download 6 zips files for a miner and 17 zip files for MAG's. After extraction to E:\papers, there are 17 Aminer JSON files and 50 mag JSON files. I group the JSON into 10 JSON files per group.

ELK stack

ELK stack includes 3 parts: Elasticsearch, Logstash, and Kibana. Elasitcsearch is used for storing and searching; Logstash is used for data collecting and pre-processing; Kibana is used as a visualized dashboard.

First, we let ELK stack run backstage. To check the running status, Elasticsearch in is port 9200; Logstash is in port 9600; Kibana is in port 5601. Second, I create a mapping schema for the academic papers. For each JSON key, I assign a proper data type and an analyzer according to the paper's JSON object example. I create 7 empty databases, as know as indices, in Elasticsearch, matching 7 groups of JSON files. Third, I create 7 logstash conf files to load 7 groups of JSON files into indices accordingly. Lastly, I can perform a search query in Kibana Dev Tools.

In week 6, I tried to optimize the mappings and Elasticsearch configuration. Because the search time for a nested field is unacceptably long, about 20 seconds. I come up with two solutions: 1. Scale the Elasticsearch cluster horizontally. 2. Optimize the mapping schema and Elasticsearch configuration. To scale the Elasticsearch cluster could be the best way to improve the search efficiency and benefit the future maintenance. I create another virtual machine (es-w2) on Azure with exactly the same settings as es-w1. However, two virtual machines can never connect to be a cluster. In week 8, I tried solution 2. Previously, I only mapped the necessary field such as title, author, and abstract. I delete the origin indices and reindexed them all with more fields, for example, fields with 'id'. Then I change the heap size of JVM to 1GB/32GB. The search time decreases significantly from 20000ms to 5000ms, sometimes even 1000ms.

Python Elasticsearch Client

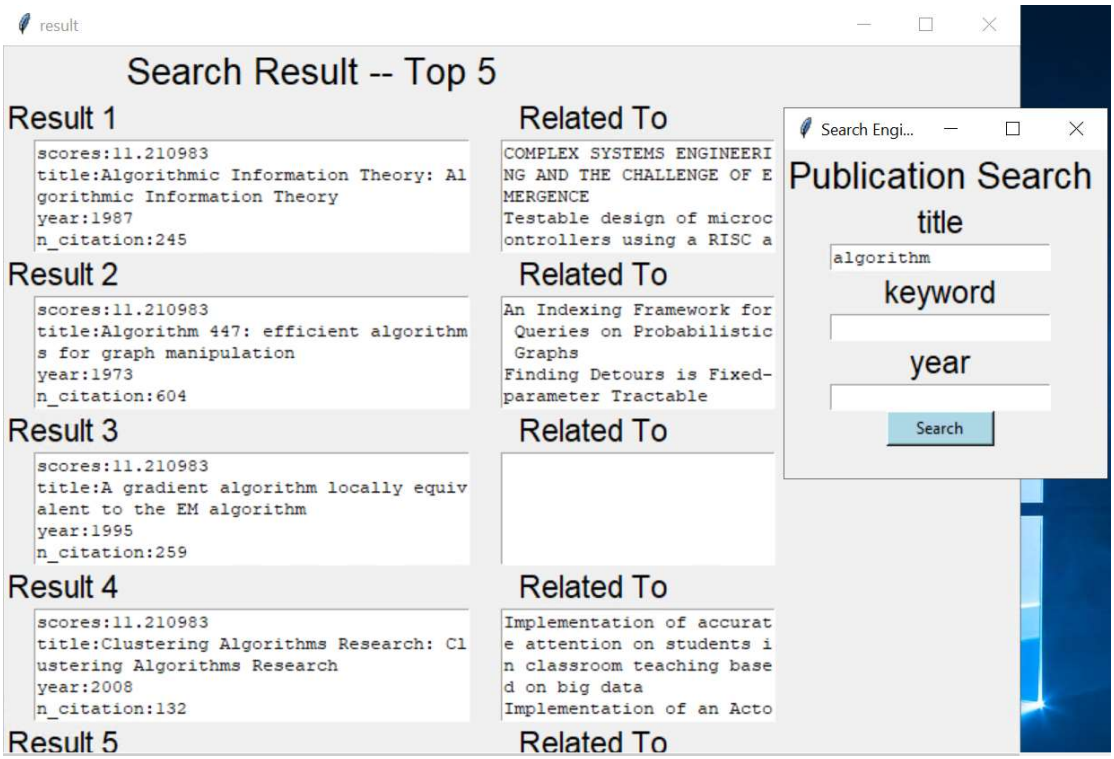
Once the ELK stack is finished. I develop an academic search engine demo using python Elasticsearch Client. It allows me to design an Elasticsearch user interface using python. There are three functions:

1. Default search:
   It is used for text search, which take an input of text, stemming and analyzing, and returns a list of papers. By default, it only searches in the title.
2. Muti-field search:
   A 'title_only' flag lets the user search on multiple fields includes 'title', 'abstract', 'indexed_abstract', 'keywords', and 'fos. name'. Besides this flag, it also allows users to see the result in a range of years. By default, it searches for the papers from 1950 to now while the user has the option to define the 'start_year' and 'end_year'.
3. Author search:
   Author search let user search on papers' author, co-author, venue, and the publisher.

The top 10 result papers are listed in descending order by the customize score. The number of citations is weighted twice as much as the relevance score (tf-idf socre).

Results

The result is shown in a graphical user interface.



Search for word ‘algorithm’ in ‘title’

Reflection

The goal is to build an academic text search engine. The first four weeks for me is a trial-and-error process. There are two popular search engine options to choose from for development. I choose python whoosh at the beginning. Whoosh is a light-weighted, pure Python search engine library. It is easy to use and implement. I selected 3000 records from the paper's datasets as a sample and test their performance. The result is not desirable, specifically, the 4 hours schema mapping time. So I switch to ELK stack then.

Schema mapping in Elasticsearch is also a big problem. Elasticsearch allows users to decide which fields to index in order to reduce the mapping time. In the first demo, after survey the paper JSON objects, I only index the necessary fields, that is, the fields including real context. Usually, the contexts are long and should be analyzed. It took 5 about hours for mapping each dataset. However, a graph search function (PageRank) was going to integrate into the text search engine. So the fields that include ‘id’ should also be mapping into the schema.

I read many papers about how the Google Scholar ranking algorithm works. Those papers show that the citation number is the most important factor in ranking. However, since we are using static datasets download from the Aminer and MAG, the citation number will not change unless we reindex everything in the future.

Future Plan

There are 3 things to do:

1. Integrate the graph search algorithm into the text search engine.

2. Design a graphic user interface.
3. Reindex the datasets once the latest datasets are released.

For more detailed progress, please refer to this link:
https://docs.google.com/spreadsheets/d/1UthP051In-S8xOmw6MXXoxBUwH58Td9W73qwGej8yi8/edit#gid=2020350960