Course Project Final Report

Tony Mu, Yang Pan and Wenjun Peng

CS410, Fall 2021

1 INTRODUCTION

There are existing job search engines that provide similar functionalities such as Indeed and LinkedIn, but they are not specified in any fields and therefore tend to provide overwhelming and irrelevant information. To complement the gap between these search engines, we successfully implemented a "FAANGMULA Job Search Engine" from scratch. This search engine limits to job searches in the nine tech companies, namely Facebook, Amazon, Apple, Netflix, Google, Microsoft, Uber, Lyft, and Airbnb. It is also lightweight and can be run from limited resources (for example, your desktop, or a free tier of Azure instance). Moreover, it is also **open sourced and highly customizable**. This means you may be able to freely extend features such as NLP, enrich data, or fine-tune the retrieval function. It solves many problems with existing job search engines and provides direct help to people we are interested in tech positions.

2 METHODOLOGY

To implement FAANGMULA job search engine, we broke down the project into three layers: the data ingestion layer (job scrapers), the application layer (indexer, ranker, and server), and the frontend UI layer. We will go into details on each of these layers in the next sections.

2.1 DATA INGESTION LAYER

We created scrapers in Python for all the nine FAANGMULA job sites. Each scraper outputs two line-separated files: a file that includes job URLs, and a file includes job descriptions. The speed of a scraper is scraping 100 job listings per minute on average on our laptops with 50 Mbps internet speed. We get about 2.9k job listings in total from the nine companies as our dataset. After successfully scraping these 2.9k job listings in nine files, we manually combine them into one data file to be used as the data for indexing and searching.

These scrapers are run offline with a schedule, preferably at least once per day to ensure our data's freshness. We also throttled our RPS (requests per second) to be one request per second, so as not to disturb normal traffic, or put unnecessary load onto these career sites.

2.2 SEARCH APPLICATION LAYER

The search application layer can be further broken down into three sub-layers: the indexer, the ranker, and the web server. We will go into more details in this section.

2.2.1 Indexer

We index the data using metapy in Python and store the index files in a temporary folder in the same directory as the indexer script. The indexer job should be run in accordance with the data ingestion jobs, to ensure index freshness.

2.2.2 Ranker

We used Okapi BM25 as the retrieval function/ranker for the search engine. It output a ranked list of relevant job links based on the job descriptions. We used explicit feedback to tune the parameters until the ranking is satisfactory based on our standards.

2.2.3 Web Server

We used a lightweight Flask web server to handle both application logic and web logic. Namely, it parses the query out of user requests, invokes the ranker's ranking function, and get compose a response with a list of job links relevant to the query.

2.3 Front end integration

Lastly, we implement a front-end view that takes user queries and displays the list of relevant links using React. We add a detailed set-up document in our git repository for users who want to try this search engine.

3 Results

We use the techniques that we have learned from CS410 to successfully create a satisfactory search engine targeting FAANGMULA job sites based on user query. The search engine provides job links directly to the sites that contain the keywords from user input. We have tested the search engine and evaluated the outcome by explicit feedbacks from users.

4 Software Demo

https://youtu.be/8gNv4SxQtSo

5 Future Steps

We had more great ideas that could further improve this application. However, due to limited time allocation for this project, we had to limit our scope to delivering a minimal viable product. Nonetheless, we are very much interested in working on the following items in the future:

1. Deploy the FAANGMULA Job Search Engine to a production environment, with potential usage of Kubernetes and Docker.
2. Grow our dataset (corpus) possibly to scale of 100,000s of job links.
3. Swap our search application layer with Apache Solr.
4. Improvements on UI with better design and color scheme, possibly a logo.
5. Better error handling on the web server for a more resilient service.