Jobs for International Students Search Engine Final Project for IST 441

Ismael Diaz
Applied Data Science
Pennsylvania State University
State College, Pennsylvania, USA
ibd5043@psu.edu

Carlos Molina
Industrial Engineering
Pennsylvania State University
State College, Pennsylvania, USA
cxm76@psu.edu

Rommel Silva
Information Science and Technology
Pennsylvania State University
State College, Pennsylvania, USA
rks5335@psu.edu

Jamin Moon
Information Science and Technology
Pennsylvania State University
State College, Pennsylvania, USA
jby5073@psu.edu

ABSTRACT

The following paper describes the team's project to build a specialty search engine for its client Dr. Robert Voigt for its IST 441 class. The procedure was as follows: the program Scrapy was used to crawl and scrape ZipRecruiter, python code was used to filter out the desired links from the websites scraped, Elasticsearch was used to index the scraped websites, and Django was used to make the user interface.

The search engine was compared to a google search engine with similar seed links and it performed favorably and the final product pleased the client.

KEYWORDS

International Student, Job, Search Engine, Visa, H1B

ACM Reference format:

Ismael Diaz, Carlos Molina, Rommel Silva, Jamin Moon. 2020. Jobs for International Students Search Engine: Final Project for IST 441. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

Introduction

Often it can be a big challenge for international students to find appropriate job opportunities that provide visa sponsorships. But with the help of our search engine, it will help relieve student's stress by being able to utilize an effective way of searching for job opportunities in the United States. Our group's main goal is to produce a search engine intended to help Penn State and other international students find job postings for internships, co-ops, full-time positions with companies willing to sponsor a visa. Our customer was Dr. Robert Voigt, who is a graduate coordinator from the Industrial and Manufacturing Engineering (IME) department at Penn State.

Figure 1.1 Dr.Voigt

1. Motivation

Our project of creating the custom search engine is

important because International students are currently in an unsettled situation where it is harder to find a job position that provides visa sponsorship. The Penn State University students are very diverse in terms of ethnicity and nationality as there is a big pool of students who are international looking for job positions. This custom search engine will be an effective way of helping international students to gain opportunities for job posting in the United States. Furthermore, our targeted audience for the custom search engine project is international students who are non-residents of the United States in need of opportunities for jobs in the country because there are currently no other ways that can provide it. It is very hard for these students due to the difficulties of filtering out jobs that require work authorization in the US. Regarding the development of our custom search engine, our biggest competition for this project would be Fastweb. Fastweb allows the users to create a profile that states whether or not if they are a US citizen, US Permanent Resident, or none of the above depending on their current status and it can overlap with the purpose of our custom search engine.

But first, what are search engines and their importance to it? There is a huge amount of information and data because computers all over the world are connected to the Internet. With the development of Internet technology, anyone and everyone can have access to a vast amount of information. At this time, the users can do a "search" to find the data that they need and the data that they want, and it's the "search engine" that makes it possible to provide the type of search the users need and perform these functions. Through these search engines, you can simply find the information you want to get by typing in a few words. Most search engines are often equipped with programs called search robots to collect the latest information. Search methods include finding them according to the classification by subject and entering keywords related to the information you want to find.

The search engine operates in three stages: Web crawling, indexing, and searching. The web crawling process copies every page of the site. This is done with the latest prioritized data and subsequent indexing processes to collect, store, and analyze data to ensure accurate and fast searching process. The final process, "Searching," extracts titles, topics, and many more other factors that contain words that make up a search term. These series of processes lead to a search itself.

Moreover, crawling or scraping is the act of taking a web page as it is and extracting data from it and the crawling software being used is called a crawler. Similarities are essential in search engines, which are used to automatically search and index various information on the Web. It is also called Spider, Bot, and Intelligence Agency. Instead of searching for information on the site one by one, the users constantly find and aggregate new web pages according to the pre-populated way of computer programs, and use the results to find a new information and add indexes to it. Although it has the characteristic of searching vast amounts of data, it also has the disadvantage of manipulating rankings or avoiding searches by using the robot's search function. A great example of this is Google because it also operates using the bots.

Regarding crawling, Python is the main programming language used in the field and is rapidly developing as libraries develop to make it easier for people in the humanities and statistics fields to use, and also for non-technical users who are not familiar with computer programming.

Searching on the Internet is about finding materials scattered across numerous web pages to suit the user’s needs. This requires a “search engine” with a variety of features to quickly and accurately find the information the users desire. For this purpose, filtering is a typical feature of the search engine in which it is a technology for filtering data. It filters data according to the standards or requirements that are required by the user, by selecting the data, and showing the filtered data.

For example, materials are divided into blogs, sites, personal pages, encyclopedias, social media, videos, photos, illustrations, books, and many more depending on their personality. Also, data search can be further refined through the length of time the data is registered, the level of filtrations that match the search terms, and the classification by country. These search engines differ slightly from one another in how to find data. For example, the world's most popular search engines are "Google" and the oldest search engine is "Yahoo."

2. Procedure

First, we crawled ZipRecruiter to get the URLs. The following URLs were then scraped, and their contents were stored in JSON format, used for the Elasticsearch database. In order to index our data, we used

Elasticsearch, and then Django was used for our UI and final webpage.

Not everything was smooth sailing at this project, and in this section, the challenges we faced along the way as a group was touched on. The first problem addressed was finding the right place to crawl. The second problem addressed and the biggest one of the project was the one concerning filtering the jobs that are not of relevance to international students. Finally, the minor changes which had to be done to the user interface in the HTML front end and python backend were addressed.

2.1 Which Sites to Crawl?

During the early stages of our project, we need to determine which sites would be appropriate to crawl according to the robots.txt. Some of the sites that we considered crawling were ZipRecruiter, Indeed, Glassdoor, Nittany Lion Careers, and Monster. However, job posting sites other than ZipRecruiter could not filter the jobs that do require visa sponsorships. In conclusion, we were able to determine that ZipRecruiter was the best site to crawl after analyzing the users allowed and disallowed. Once we determined that ZipRecruiter was the best fit, we crawled over 74,000 active URLs that included internships, Co-ops, and full-times. However, we quickly realized that many of the URLs were expired and required sponsorship.

2.2 Filtering Keywords

International students have a hard time sifting through all of the job postings available to find the ones they can actually apply to. This issue is the main problem we want to solve with our search engine so our main objective throughout this project was to serve that function. The best idea we came up with looked to solve the problem after the crawling and before the indexing by filtering the unwanted jobs before they were indexed. This decision was made because the other steps where it could have been filtered were not as good options. There wasn't any website just full of job postings we could crawl (if there were our project would have been useless) so the filtering had to be done after that. The bare links couldn't have been crawled because they did not show anything in the body which could be scrutinized to filter them. Finally, after indexing the data was harder to modify, leaving the step after the crawling and before the indexing to be the best option.

Because of the way our search engine was set up, this step happened to be implemented by filtering the JSON lines file. A python code had to be written which could take in data.jl file Two different approaches were considered: an exclusionary approach and an inclusionary approach.

The exclusionary approach consisted of excluding the links that were definitely not useful jobs, while the inclusionary approach consisted of only including the links that were definitely useful links. The main difference between both approaches is the question of including the "ambiguous" links which don't specify if they allow international applicants. After consulting with many international students and amongst ourselves, the team decided to include these links because of two reasons. First, these links comprised the vast majority of links (~80%), therefore, eliminating this group would have significantly reduced the number of links we had. Second, many of the job postings which have no explicit mention of allowing international applicants actually allow for them.

Once we decided on the exclusionary approach the team went to work on looking for key phrases which indicated that the job posting referred to a job that was only available for US Citizens. This was made by studying the postings which appeared in our Google CSE and looking for those phrases. The final list we came up with which seemed to be a good filter (after many uses of our search engine we haven't found a single job which explicitly disallows international applicants) is below.

Table 2.1 Exclusion Filters

'require sponsorship'
'Without the need for employer sponsorship'
'Without the need for\nemployer sponsorship'
'without sponsorship'
'without needing sponsorship'
'employment authorization'
'authorized to work in the U.S'

The next step required the team to filter out the job postings which had any of these key phrases. This was done using the following python code shown below:.

Table 2.2 Filtering Code

```
# -*- coding: utf-8 -*-
```

```
import json, os, sys, getopt

def getArgs(argv):
        input_path  = ' '        #path of input file(s)

        try:
        opts, args =
getopt.getopt(argv,"hi:",["input_path="])
        except getopt.GetoptError:
        print 'usage: filter.py -i <input_path>'
        sys.exit(2)
        for opt, arg in opts:
        if opt == '-h':
        print 'usage: filter.py -i <input_path>'
        sys.exit()
        elif opt in ("-i", "--input"):
        input_path= arg

        return input_path

def filter_data(file):

        file_path = file
        j_content = []
        new_j = []
        filters = ['require sponsorship', 'Without the
need for employer sponsorship', 'Without the need
for\nemployer sponsorship', 'without
sponsorship','without needing sponsorship',
'employment authorization','authorized to work in the
U.S','require immigration-related sponsorship']

        num =0

        with open(file_path) as f:
        for line in f:
        j_content.append(json.loads(line))
        olditem = []
        for item in j_content:
        if '/Job/' in item['url'] and item != olditem:
        new_j.append(item)
        olditem = item
        for f in filters:
        if f in item['body']:
              new_j.remove(item)
              print(item['url'])
              break
        return new_j

def main():
        argv = sys.argv[1:]
        input_path = getArgs(argv)
        data_filtered = filter_data(input_path)

        with open('data_new.jl', 'w') as outfile:
        for entry in data_filtered:
        json.dump(entry, outfile)
        outfile.write('\n')

if __name__ == '__main__':
        main()
```

What this code basically does is first check if it is being called correctly, it then reads the json lines file that is delivered from scrapy (from using scrapy crawl urlgetjson -a filename=links.txt -o data.jl) into a list called j_content made of json lines using the json library of python. Afterwards it runs the for loop shown below:

Table 2.3 Filtering code for loop

```
for item in j_content:
        if '/Job/' in item['url'] and item != olditem:
        new_j.append(item)
        olditem = item
        for f in filters:
        if f in item['body']:
              new_j.remove(item)
              print(item['url'])
              break
        return new_j
```

The first thing checked for is if the link section of the item being analyzed is an actual job and this is done by checking if the url section of the json line contains "/Job/" which in ZipRecruiter would mean the link contains an actual job posting. Afterwards, the current json line is appended to a new list. It then assigns the current item to the variable "olditem" in order to not insert the same item twice into the new list. It then cycles through the list of filters and checks if any of the json lines contains any of the filter phrases in its body section, if so it is removed from the new list.

Finally, the code writes all the items in the new list to a new json lines file, which is the one then sent to ElasticSearch.

This code took too long (over an hour) to filter the data.jl scraped previously, which made repeated runs for troubleshooting hard. The solution that the team came up with was having an additional filtering step before

actually scraping the websites. The links.txt file was filtered similarly to how it was explained, but using a list of strings rather than a list of JSON lines and only looking for the "/Job/" keyphrase.

2.3 Personalizing the User Interface

The HTML code for the home and results websites was modified to be more personalized for the intended users. First, to fix the broken links which were shown to the user, the python backend code was modified to cut the first 7 characters of every URL (since the broken section seemed to be the "https://" section), this allowed for proper links to be given to the user.

The HTML code was changed to add a "Previous 20" and "Next 20" buttons. To add these buttons links were added to the code using the <a> tag to link to the same webpage but with the start condition displaying 20 more or 20 less than the current start.

The python backend was modified to filter out any text from the total variable so it displayed only the total number of webpages in the results page.

Finally, minor aesthetics changes were made in order to make the website more appealing. The results of these changes is shown below:

Figure 2.1 Home Webpage

3. Results

Link:
https://cse.google.com/cse?cx=006256541337229453869:teicy3ywy1x

There are several advantages of Google custom search engines and one of them is its great popularity. As they are the world's leading search engine with more users than one can imagine, Google will continue indexing more and more new information and provide it toward its users. It is almost entirely sure that Google search engines have over a trillion websites indexed already and because of that its users are finding information that they need and even a wider variety of information that is relevant to its search that the users have not even thought of. This is why Google takes the lead when it comes to search engines on the internet.

Another advantage that Google search engine has is that it provides its users with the most relevant sites at the top of its search results list. The search engine has a rating system of the sites based on the popularity and its link to the site so that it can automatically get a better ranking for the search results. Furthermore, the variety of file formats that the Google search engine can index allows the site authors to freely store files in various formats without having to worry about Google not crawling their sites.

The major advantage of our custom search engine is the ability to filter out the jobs, where only US citizens are accepted. This will allow international students to focus on relevant jobs not requiring sponsorship and increase their chances of obtaining a job. However, the database for the Google Custom Search Engine (CSE), is much larger with roughly eight million job postings, as compared with ours with only 47,000. Also, the CSE has a better user interface, with the control panel offering advanced features.

Figure 2.2 Cross Reference with MyVisa.com

Upon completing our custom search engine we decided to present it to our customer Dr.Voigt, in which he suggested a few recommendations. The first recommendation was to implement three types of categories to illustrate the certainty of sponsorship. The job postings would be tagged with the following yes, no, or maybe. Once those job postings are tagged, the second recommendation that Dr.Voigt mentioned was to implement some type of user feedback option. The user would be able to provide feedback on the accuracy of our search engine, based on their search results when applying. The final suggestion that Dr.Voight mentions are to cross-reference the job posting with the website MyVisa. By doing so, this will allow us to focus on the top companies who are more willing to sponsor international students and get further insights to improve the performance of our search engine.

3.1 Performance Comparison

In this section the precision of the speciality search engine made by the team and the Google CSE is calculated by searching the queries shown below and getting the percentage of "relevant" links of the first 10 results, the relevant links in this case referring to any job postings which accept international students.

Table 3.1 Precision Results of comparing our search engine to google

Query	**International Student Job SE (Ours) Precision**	**Google CSE Precision**
'Boeing'	90%	0%
'Consulting'	100%	10%
'Python'	90%	50%

Conclusion

To summarize the main purpose of our project was to design a search international students find jobs for our client Dr.Voigt. We were able to crawl the Ziprecruiter and index the output to develop the search engine. In addition, we also created a Google Custom Search to analyze the advantages and disadvantages. Upon completing both search engines we met with our customer Dr.Voigt to present our results, and seek recommendations. We received lots of positive feedback and interesting insights on how we can improve our search engine in the future. By implementing the following recommendations, it would make our search engine a better tool to help international students find their dream jobs.