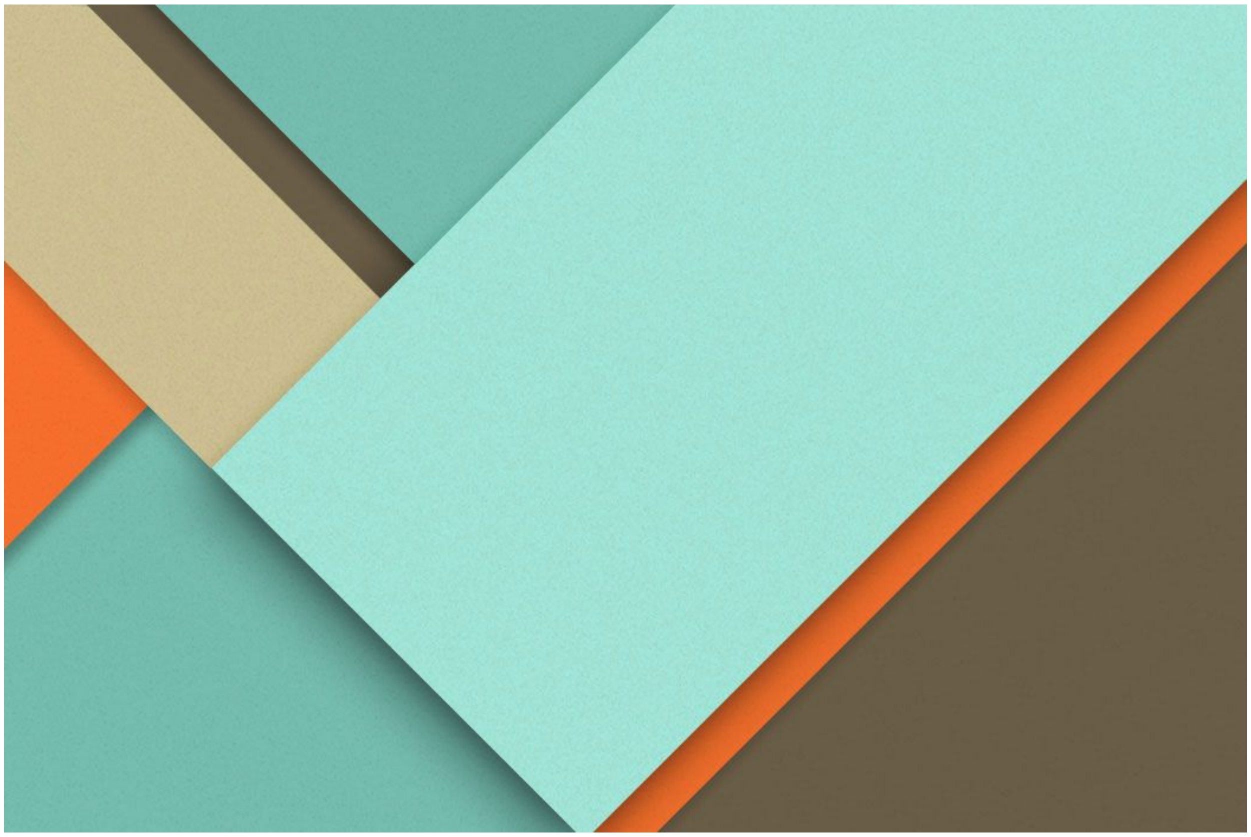

Skill Mapper

04.16.2018

—

Sahaj Arora 100961220
Jennifer Franklin 100315764
Luke Daschko 100976007

COMP4601
Intelligent Web Based Systems

Abstract

Skill Mapper is an application that crawled two job sites: Monster and Jobboom for a list of 218 skills and records the top ten associated skills for each. These associated skills are displayed graphically showing a percentage of total job sites found containing each associated skill.

The main tools used to produce Skill Mapper is Crawler4, which was used to crawl the job posting websites.

Introduction

When looking for jobs it is recommended that you find out more about jobs that interest you by going online or talking to people in the industry. When researching jobs online there are many job search sites available. In Canada some of the top sites include CareerBuilder, Eluta, Indeed, Jobboom, LinkedIn, Glassdoor, Monster, Simply Hired, and Workopolis. In fact a plethora of information is available to a job hunter, the challenge lies in sifting through the vast quantity of information that is available.

When searching for jobs one of the first questions a person might ask is “what are the related skills I need to know if I am interested in a specific job”? Our project attempts to answer this question. Provided a job title our application lists the top ten skills associated with that job title with data aggregated across potentially hundreds of search results in order to obtain this top ten list. Results for a specific job title can run into the hundreds if not thousands of hits and attempting to gain an overview of such a vast quantity of data might be daunting. Our application provides a service unique to this situation and is posed to benefit job hunters. In fact anyone who wants to stay on top of what skills are in demand should make use of our application. So if, for example, someone is good with natural language processing then we know that they want to have an edge they should have equally strong skills with respect to python, big data, java, tensorflow, javascript, keras, theano, nltk, and c++, and html.

The remainder of this report is comprised of:

Background - this section includes the background information needed for this project

Related Work - this section includes a synopsis of works related to this project

Methodology - this section includes a summary of how the project obtains its results

Discussion - this section details what worked, what failed and what the team learned from the project

Conclusion - this section concludes the report

Future Work - this section looks toward future direction the project could take

Background

In order to understand how Skill Mapper functions and runs it is important to understand what technologies were utilized in its development. The main tool used to produce Skill Mapper is crawler4j. Crawler4j is a open source web crawler tool for Java, which allows for efficient multithreaded crawling. This is what we used to crawl job posting websites. This tool was used to navigate through Jobboom.com and Monster.com.

Related Work

I. Americanjobcenter's MySkills MyFuture

The Americanjobcenter's MySkills MyFuture is a web application designed to recommend career paths and training based on an individual's current or previous job positions. The user inputs his or hers previous job title and the website generates a collection data regarding future employment opportunities available to them. The focus of the tool is comparing the job input to a similar potential job for the user, comparing education, training, salary and skills.

Similar to our project the user can gain insight into what skills they should advance in order to ensure more success in their job search. MySkills MyFuture focus on providing skills the user likely already has or skills they have been exposed to, in order to compare the users inputted job with the selected future job. Our program goes a step further by display skills the user that jobs posting request, which many times go beyond what the user has come into contact with.

In terms of data the two applications draw from (potentially) different pools. Skill Mapper (our program) scrapes data from jobs posting found on two of Canada's biggest job listing websites. MySkills MyFuture access data provided by the United States Department of Labour. Skill Mapper keeps up with what employers want to see based on what employers publically ask for. It is unknown how up to date the United State Department of Labour is with their statistical analysis of jobs, instead focusing it attention at providing data that would otherwise be restricted from the public (as much of the Department of Labour data is not readily available).

II. O*NET Online

O*NET Online is website dedicated to helping users find correlations between a job type and a plethora of related subjects. Users have the option to search for job types based on multiple fields including: skills, tools & technology and current occupation.

Unlike our project O*NET allows user to input beyond technical computing skills. O*NET search by skill tool givers users the option to input skills from different categories ranging from Basic Skills (reading, speaking...) to Social Skills to Technical Skills and more. After selecting a search tool, the users is presented with a list of job types. Once the user accesses a jobs profile they can view a detailed list of specific skills required for this job (including a list of detailed technical skills).

Like Americanjobcenter's MySkills MyFuture, O*NET was granted access to data from the U.S. Department of Labour. Although there is a benefit publicizing otherwise private data, the Skills Mapper team found it more useful to provide consumers relevant and up to date data provided directly from employers.

Methodology

From a birds eye view the project can be broken down into two core components: data mining and statistical analysis. The data mining component may be further broken down into two broad sections: crawling and regular expression analysis.

The entire process starts with a job term being chosen from a list of skills catalogued in the database. Two job search sites: monster and jobboom are crawled for the specific job term. Our application then lists the top ten skills associated with that job term with data aggregated across potentially hundreds of search results in order to obtain this top ten list.

Each site that was chosen for crawling had to meet certain criteria including:

1. A robots.txt file that allowed crawling of the URLs of interest.
2. A unique starting URL that matched a fixed predictable pattern that could be coded based on the desired search term.
3. A set of next page links where the URL's followed a fixed predictable pattern that could be coded based on the desired search term and pattern based rules for the pagination changes in the URL.

Data Mining

Crawler4j is the core technology that drives the crawl of the three job sites. Three seeds start the entire process, one for each site. Those three seeds are:

1. `"https://www.monster.ca/jobs/search/?q=" + searchword + "&where=canada"`
2. `"https://www.jobboom.com/en/job/c_canada/_k-1?dk=" + searchword + "&location=canada"`

Where searchword is the job term that has been chosen for which the top ten associated skills are to be determined. Should the searchword contain special characters, for example c++, the searchword is encoded using UTF-8 encoding, with this example it would become c%2B%2B for jobboom and then using a regex it is converted to c__2B__2B for monster as they use their own custom URL encoding.

Once the crawl is started there is "should visit" logic that looks for URLs that match the next page format of the crawl result for each of the different job sites. The first time a URL matches the next page format it is added to the list of links to be visited, thereafter it is not revisited. In other words page 2 is visited the first time 2 is matched but after 2 has been matched it is not visited again.

During the crawl the crawler can of course also visit a job posting. The crawler can analyze the URL and differentiate whether it is a search result page or a job posting page, if it is not one or the other it will not visit it. Similar to the logic used with analyzing search result pages the crawler collects all job post pages into a URL, if the URL is already in the set, it is not unique and is not visited.

The technique used to analyze URLs is simple string methods like String.substring, String.indexOf and String.contains. Although these familiar methods may seem too simple they proved to be highly efficient and effective.

Statistical Analysis

At a high level we compute our ratios using the following process: We compute a set of true/false values for every word from our skills dictionary for each job posting based on a specific skill query set of page results. The database has a table of skill equivalencies, for example pl sql and pl-sql are the same thing although we search for both. After we have counted the individual skills we take those counts that are equivalent and combine their counts and remove the redundant skill. After checking for equivalencies we then compute which words come up the most in the set of job postings by counting the number of true values of each word in the dictionary for all job posting and dividing by the total number of job postings.

Once the ratios are computed a small web interface was built that allows users to interact with the skills list and associated skills results. The web interface has two pages the first a drop down list from which a user may select their skill of interest as seen in Figure 1. The second page is a chart showing the top ten relevant skills associated with the skill of interest as shown in Figure 2.

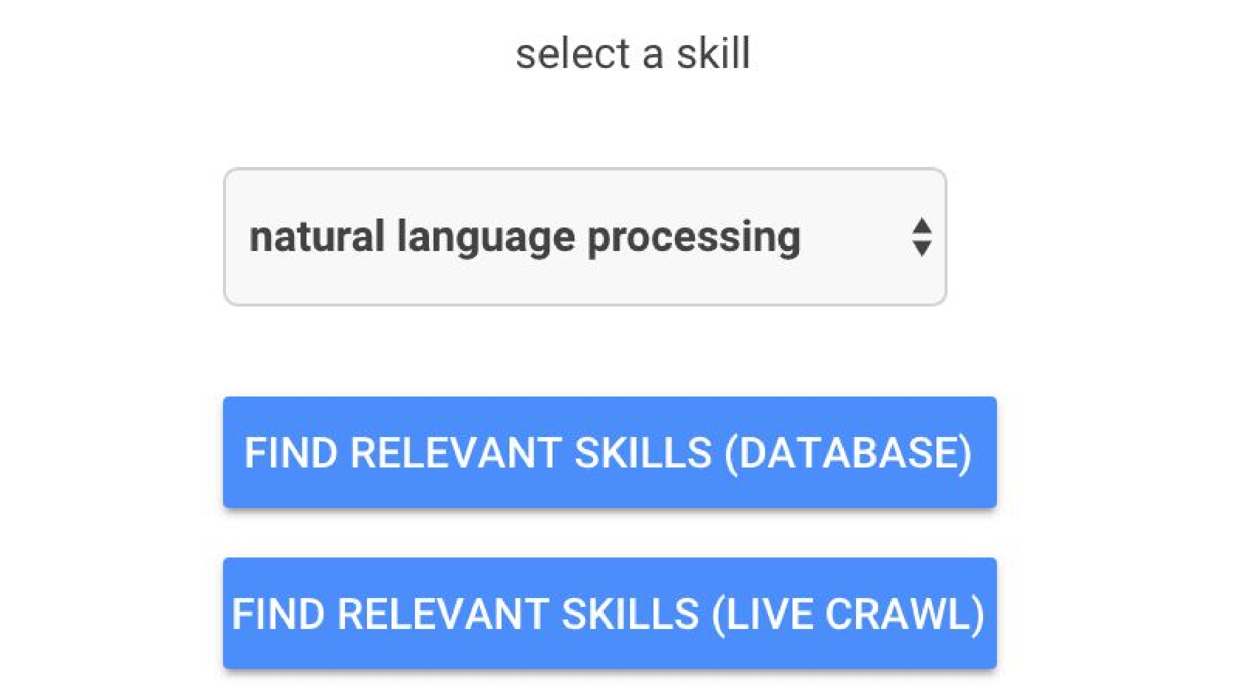
select a skill
natural language processing
FIND RELEVANT SKILLS (DATABASE)
FIND RELEVANT SKILLS (LIVE CRAWL)

Figure 1 - Selecting a skill

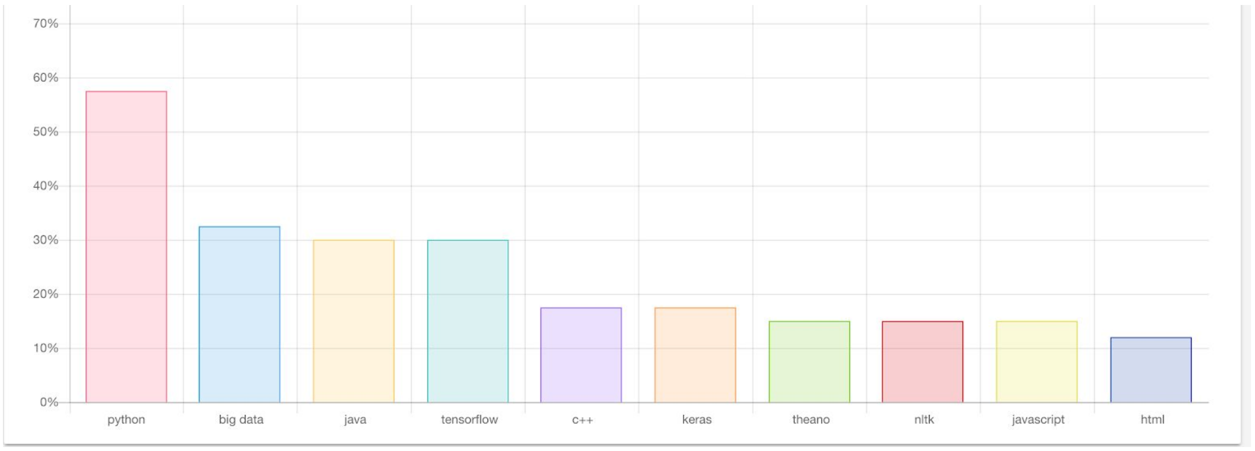

Figure 2 - Chart with top 10 relevant skills

Discussion

What Worked

Our project successfully was able to crawl two different job listing websites, collecting postings exclusively relevant to a specified term, and using a dictionary of terms, produces what skills are related to the inputted term.

What Failed

During the development process the development team faced many ups and downs and discovered the true nature of coding with the web. Originally and for most of the development process, our team had a third crawl site fully working: Workopolis.com. This was the first site coded to work, but as development went on the website itself underwent changes that made it impossible to work with. The first change Workopolis underwent involved its HTML dramatically changing, this made grabbing the needed job posting URLs a challenges as new URLs were added which made the crawl mistakenly search unwanted pages. Once this feature was fixed, Workopolis changed once more by exporting almost all of it's job pages to Indeed.com or the websites of the employers. This last change is what fundamentally forced the development team to drop Workopolis from the crawler.

Two other sites that were meant to be included were Indeed.ca and Eluta.ca. The development team discovered that both sites robots.txt disallowed all actions on their respective websites (more about robots.txt files will be discussed in the What The Team Learned section bellow). So instead jobboom.com was utilized as it provides more flexibility when crawling.

A third site that was evaluated for inclusion in the project was glassdoor.ca. However, the development team was unable to discern a predictable starting seed for crawls on the site as the URL had some sort of generated number at the end of each URL that we were unable to replicate.

What the Team Learned

Coming away from this project the development team has learned the adversities that come with web development. Perhaps the most prominent discovery was the reality of how some websites implement robots.txt files that disallow all behavior. A robots.txt file is a file associated to websites, instructing web robots (usually crawlers) how to browse the websites. Unfortunately our team discovered the Indeed.com robots.txt disallows all crawling potential.

Our development team discovered the importance in multithreading in crawling and learned to utilize theses features in crawler4j. In early models of the project, search result pages were crawled and then upon visiting these pages each job link would be visited using JSoup and then parsed. Although this allowed for clever use of regex expression to acquire job links with incredible accuracy, it also under utilized the multiple threads in crawler4j. Rather than distributing the scraping of data of the 20+ job sites per search page among the 7 threads, the previous design would commit all this work to a single thread. This created incredibly longer cawl times. After understanding the importance of the multithreading and developing a method to utilize it, the crawl dramatically improved in time.

Throughout our crawling development our team also had the the opportunity to see how unsanitary the web can be. Our team assumed professional websites like Monster and Jobboom would have polished, consistent, and organized html pages. Our team quickly realized that this is not true, many times websites have inconsistent, sloppy and convoluted html representations for their webpages and that is why it is incredibly important to have accurate scraping code that avoids collecting unwanted data.

Furthermore we also learned that certain websites do not need to follow conventional url encoding patterns. While JobBoom use UTF-8 encoding pattern, Monster seems to use a url encoding pattern that was unrecognizable (possibly a custom url encoding). Monster had UTF-8 encoding but used double underscores instead of percent symbols. This caused issues until our development team finally understood this. Once more this highlights how inconsistent and unorganized the world of web crawling can be.

Conclusion

An application that crawls three job sites: monster and jobboom was created. It will crawl these three sites for a list of 218 skills and show the top ten associated skills with a percentage of job sites found containing each associated skill. A database crawl can take up to four minutes per skill. A live crawl may take as long as two minutes to complete. These results are only as accurate as the sampling of job documents that are taken which may vary with each crawl. We are hopeful that the results will be found useful to job seekers.

Future Work

Given more time our development team would improve Skill Mapper to work faster, more efficiently and provide service to a wider clientele. A key way to grow Skill Mappers clientele would be to expand the available terms to search as well as expand the terms that can be registered as skills. This can either be done by hand or more efficiently with an intelligent system to crawl and using natural language processing, highlight patterns in language that can be used to flag down and record a job postings listing of skills. By expanding, the Skill Mapper can eventually be used by not just eager job hunters in the computing domain, but can expand to all and any work domain.