Personalized Job Matching

Md Mustafizur Rahman
mr4xb@virginia.edu

John Clougherty
Jpc3ap@virginia.edu

Sam Hewitt
Ssh5d@virginia.edu

Elise Clougherty
Emc8tq@virginia.edu

ABSTRACT

In this era of information technology, finding a perfect match between the job seekers and the job posters is very difficult. People searching for a dream job often find themselves in a place or position they do not like. Similarly, companies are always in search of productive, skilled worker. This creates an emerging need among the people and the companies for an automated system which can dynamically recommend suitable jobs for applicants and candidates for companies. Our project, Personalized Job Matching, tries to find an automated answer for this problem. In this project, we use a crawling mechanism to collect both historical data and recent data of job postings. From this data, we develop a model that receives resumes from the candidates and requirements from the companies as inputs, builds up index using some advanced natural language processing and retrieves the related jobs or potential candidates for them. Additionally, we have analyzed the underlying dynamics of jobs and people's motivations towards jobs as a byproduct of the system.

1. INTRODUCTION

Finding a job in today's market is a significant challenge. The scarcity of available jobs combined with the automated and impersonal nature of online recruiting require people to turn more and more to their personal network to find work. Precedence of friend and family referrals has become not only common but also heavily integrated into the standard corporate hiring processes at popular companies such as Google. Recruiters at school job fairs tell potential applicants to apply online despite knowing that they lack a system to identify well-matched applicants. This automated process limits the ability for personal connection and identifying well-matched applicants.

Since today's job search engines show only current listings, job seekers are prevented from seeing all of the companies that employ a given position in a certain geographical area. Once the listing is taken down, the company becomes "invisible" to the job seekers. LinkedIn promises to facilitate job placement by exposing the companies that employ one's friends and family. The limitation of this system is that the job seeker is not aware whether these companies hire the desired position. Clearly the current systems available to find jobs do not provide as detailed or complete information as some job seekers desire. There are also few tools available to statistically match an employee to a job or a job to an employee. Here we have built a personal job database that empowers job placement by crawling leading corporations, such as Google and Facebook, to build a data set of job listings overtime for a specific field of interest. Users on both the applicant and hiring side would be able to query this database to find jobs and potential hires.

The collected data is comprised of the set of jobs that are available now and were available in the past. This data of historical job postings allows a user to make a more precise decision about which company to apply to in two ways. First, the applicant can decide if he or she would be willing to wait for a desirable job to become open given the historical trends of past jobs. Second, an applicant could know more about what type of jobs the company offers in general that may be available for transfer or advancement. Similar to the way some travel sites predict fare increases, our job site will be able to predict and report on hiring trends, further empowering potential applicants.

When an applicant wants to use this system to find a job they would upload personal information about their skills and experience as a resume or HTML form. At this point, they could also specify any constraints to the jobs that they would want returned. The system would read and parse the resume and then return job postings ranked by relevance. The user could be shown how good of a match they are to each job, and possibly more importantly learn which skills or experiences they are lacking for jobs that they desire. Also, job providers could get a set of skilled worker for a specific job from our system. The benefits of such a system would be enhanced if our product could participate in the new wave of application-to-application data sharing so we could be granted permission to expose the data between systems. In this case, sharing of the personal data listings would require complete access to the job seekers LinkedIn data, Google contacts or alumni databases. It is becoming imperative that companies who own a monopolistic share of social data open up their systems to data-sharing between synergistic applications. For example, LinkedIn offers API's that access an individual's user data if given permission by the user.

In summary, the major contributions of this project are 1)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

a database consists of jobs, companies and job related information collected by crawling from major job related sites, 2) a noble job and candidate recommendation system, 3) in depth analysis of underlying jobs and companies dynamics.

The rest of the paper is organized as follows: in section 2, we discuss about some traditional job search engine and related research work in this sector. Problem definition is discussed in section 3. Section 4 describes the heart of our system in details, which is followed by evaluation of our system in section 5 and some additional features of our system in section 6. Our findings of job analytics are presented in section 7. Finally, we conclude in section 8 with some direction of future works in section 9.

2. RELATED WORK

During this time of information overload, the need for a perfect recommendation system is crucial for job seeking and recruiting web sites. But most of the current web sites largely depend on the keyword based search. However, simple keyword-based search and filter techniques cannot be sufficient to capture the underlying job dynamics. To improve the traditional systems, content based filtering or collaborative filtering can be applied.

In content-based recommendation one tries to recommend items similar to those a given user has liked in the past, whereas in collaborative recommendation one identifies users whose tastes are similar to those of the given user and recommends items they have liked. In [4], the authors use collaborative filtering to recommend jobs to users. However, the main problem of this system [4] is that it is critically dependant on the availability of high-quality user profiles which we cannot expect in the real scenario. Hybrid system, combination of both content and collaborative filtering are also explored in [1]. In [3], the authors propose a recommendation system where they incorporate both content and user actions using the graph node analysis in the job seeking and recruiting process. However, the fundamental shortcomings of these methods are that these are highly dependent on the user interactions (i.e. like) and it is really hard to like or recommend a job for a user who has never had that kind of job.

Now if we take a look at the leading job search engines, Glassdoor and LinkedIn, we will find that they only allow for traditional keyword based search to query their list of jobs. On their platforms it is impossible to get a list of jobs that is personalized to your set of skills and experience. They allow filtering by company and location, but it is still up to the user to find jobs that they are actually qualified for which limits the efficiency of the search. The Glassdoor displays the simple keyword search and also shows ads from companies looking to promote their job listings. Despite being tied to a users account or Facebook, there is nothing personalized about the search results. LinkedIn's search pages shows some amount of personalization but it is all determined by the content of a user's past searches rather than the skills that they are allowed to input into the system.

In our work, we try to incorporate content based analysis on the user profile and job recruiters' requirement (experience oriented keyword search) using the historical log of job postings from the companies. To the best of our knowledge, no previous works in the literature has ever used the historical job postings.

3. PROBLEM DEFINITION

One of the fundamental focus of our proposed system is to facilitate the experience oriented keyword based resume and job search. This involves taking into consideration the users' experience level as well as expertise. Same is true for the job postings with specified experience level and responsibilities. We try to rank the resumes or job postings in the higher position which have better relevance with user expectation in terms of their expertise and experience level.

The second important contribution of our system is the job recommendation based on resume similarities. Most of the modern systems recommend job based on collaborative filtering or content based filtering does not taking into consideration what is the credential of the users. To tackle this issue we envision a job recommender which solely lies on the similarities between the resume. The intuition behind this is that people having similar credential might be eligible for similar job.

4. OUR APPROACH

We sought to build a more robust information retrieval system for the job market. As most current systems operate on a traditional keyword based search, our goal was to make the job search more personalized to the candidates and positions that were seeking a match. In this section, we introduce the methods we have developed to build our system in more details. As a specialized search engine our system contains the followings part.

1. Crawler: Four specially designed crawler to continuously collect data from four different job and resume websites.
2. Data-set parsing: Parsing the data-set of the resumes and job from the crawler into suitable format.
3. Document analyzer: One of the fundamental and core component of our proposed system which perform some key steps of our proposed system. It analyzes the parsed data in more depth and finds more meaningful meaning of the data.
4. Indexer: Perform indexing on the analyzed documents.
5. Ranking and Retrieval: Perform search and retrieval to provide ranked results.
6. User Interface Design: A suitable user interface for the easy interaction.

4.1 Crawler

Crawler for any search engine system is the main part for feeding data to the system. The main job of the crawler is to crawl different websites and store the crawled data into suitable format. As a specialized search engine system for job and resume search, our main focus on the job postings and resume websites. While performing the job of crawling, we found that although job postings is quite available it is really hard to find any online resource for free resume.

After some careful inspection, we decided to crawl the job postings from the three major computer science job employer namely: **Google**, **IBM** and **Facebook**. Although initially our decision was to crawl personal profile or resume from **LinkedIN**, but later we have to abandon this idea because of privacy issue. So finally we decided to use **Indeed.com** to crawl some resume related to the job like *Software Engineering*, *Data Scientist*, *Network Engineer*.

4.2 Data-set Parsing

From the discussion of Crawler, it is clear that our data-set consist of job postings data from three different websites and resume data from one website. Since most of the data are in free text format, we have to perform a significant amount of preprocessing to store the data into suitable format. It is to be noted that we have used JavaScript Object Notation (JSON) to store the data after we perform the data-set parsing on the crawled the data.

Each job posting JSON file contains the followings fields.

1. Job Title
2. Location
3. Responsibilities
4. Minimum Qualification
5. Preferred Qualification
6. Company
7. URL

Other than this we also store the job department, posting date, hiring date, salary etc. Similarly each resume file contains fields like Name, Tile, Educational Qualification, Experience, Additional Information, etc. It is to be noted that our data-set consists of around 6000 job postings from Google, Facebook and IBM; on the other hand we have around 1450 resumes from indeed.com

4.3 Document Analyzer

Once we have all the data in suitable JSON format, our next phase is to analyze these data carefully and try to find out more meaningful pattern from it. Document analyzer has several important phases which are as follows:

1. Stopword removal
2. Stemming
3. Automatic Keyword identification
4. Keyword oriented experience correlate

4.3.1 Stopword Removal

Much like traditional document retrieval, the information in the job listings and resumes is not in the optimal format for IR retrieval methods. The traditional list of stopwords is not sufficient for job listings since there are words like "qualification" or "requirement" that are very prevalent in this corpus. Therefore, we generate a custom list of stopwords to remove before sending information to the indexer.

To do so, at first we remove the traditional stopword using the smart system stopword list. Then we simply calculate the frequency of each word across our entire corpus. Once we have the frequency of all the words, we sort this list according to the frequency in descending order. Figure 1 illustrates the stopword list compiled from Google Jos Postings. From this it is evident the its clearly following the Zip's Law. We discover a separate set of stopword (Table 1) for Job and resume like the word *method*, *profession* etc.

Table 1: A small subset of the stopword from Google Job Postings

Word	Frequency
Drive	1446
Analysis	970
Solve	432
Reliable	319
...	...

Besides, there are also many abbreviations such as M.S. (Master of Science) that need to be expanded to their full forms through dictionary expansion in order to meaningful for retrieval.

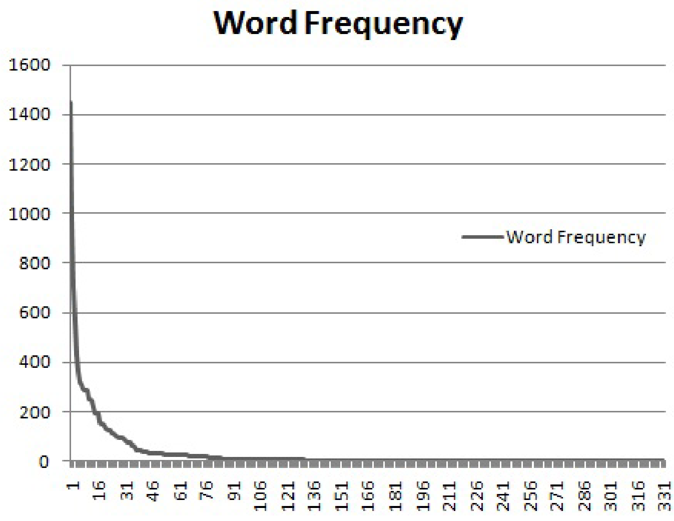


Figure 1: Word Frequency Plot of Google Job Postings

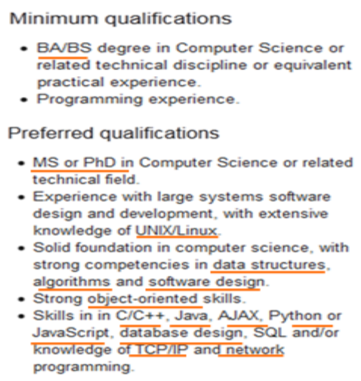
4.3.2 Stemming

For the purpose of stemming we have used the standard stemmer provided by the Lucene package.

4.3.3 Automatic Keyword identification

Since one of the most important part of our system is the experience oriented keyword search engine, this is one of the most important phase of document analyzer. Before going into further details let take a look at the Figure 2, one of the job postings from Google. From the figure 2, it is clear that the set of words a job seeker usually look for from a job postings. We can simply categorize these words into three groups, a) qualification, b) expertise in different sections and c) corresponding related experience.

Although this type of identification is very easy for human, searching for these three categories automatically from free form of text is really very difficult. Using simple Unigram based model, will loose some important keyword terms like "software design", we need this as a whole, but in Unigram model we will get "software" and "Design" separately. Phrase Query might be a possible solution. But the problem is the usually the fields like "Qualification" are very large to handle using phrase query. Nevertheless, none of these methods can automatically identify the keyword set.


Minimum qualifications

- BA/BS degree in Computer Science or related technical discipline or equivalent practical experience.
- Programming experience.

Preferred qualifications

- MS or PhD in Computer Science or related technical field.
- Experience with large systems software design and development, with extensive knowledge of UNIX/Linux.
- Solid foundation in computer science, with strong competencies in data structures, algorithms and software design.
- Strong object-oriented skills.
- Skills in in C/C++, Java, AJAX, Python or JavaScript, database design, SQL and/or knowledge of TCP/IP and network programming.

Figure 2: A screen shot of Google job posting

Table 2: A small subset of experience oriented keyword list from Google Job Postings

Keywords	Experience
C++, Java	2 years
Software Design	5 years
TCP/IP	Null
...	...

To solve these problem, we envisioned to use natural language processing. From the figure 2 we can develop some sense of intuition of how to identify these keywords and where to locate those as follows:

1. Most of these keywords are Noun.
2. Most of these keyword appears only after some preposition (in, with).
3. For multiple word keyword (i.e. Software Design) search for consecutive Nouns.

So we simply use the parts-of-speech tagger and apply the above intuition in algorithmic manner and get very promising results.

4.3.4 *Keyword oriented experience correlate*

The next important thing is to identify the associated experience a job seeker have or the posted job is looking for. The basic things is that we are looking for something as illustrated in table 2. So to obtain this information we again take the help of natural language processing. At first for each different automatically identified keywords, we search in which line it occurs, then we simply search for a number associated with either word "years" or "experience" or both.

4.4 Indexer

We built two different indexers for the job postings and the resumes. Job postings were indexed on the following characteristics: title, location, qualifications, responsibilities, and experience oriented keywords. The experience oriented keywords come from the processing described in the section 4.3.3 and section 4.3.4. Resumes were indexed on the following: title, educational history, work experience, and skills. We also ran resumes through the same process to get matching experience keywords. Jobs and resumes that have matching keywords will receive higher relevance scores during the matching process. All the fields are string fields.

4.5 Query processing

To simplify query processing the system provides two distinct interfaces for querying information. To search for a job, the job seeker enters the fields for educational experience, job experience and skills from his resume into our system. To search for an applicant, a company enters required qualifications, responsibilities, and description. The figure 3 shows how the fields entered are queried against each other during retrieval process.

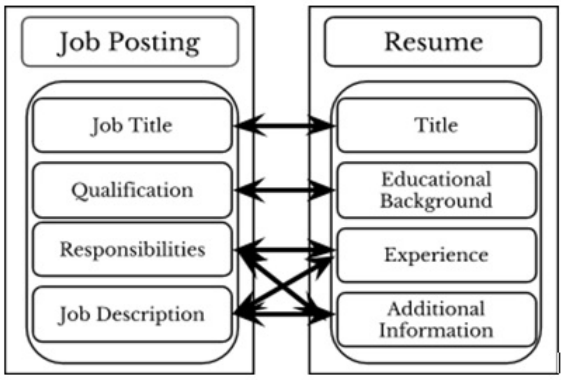


Figure 3: Matching against the fields

It is important to note that since a query itself looks like a document (either resume or job posting) we have to perform same analysis as we perform in the document analysis part. An important decision we made during the design of the retrieval system that a resume matched with more number of experience based keyword set are more relevant than a resume simply matching exactly with the "job title". Thus, the decision of boosting job postings or resume documents was made based on the number of experience based keywords its matched with.

4.6 Ranking and Retrieval

For retrieval we consider matching job title, qualification, automated keyword set and the experience correlation from the index. We tried two methods of retrieval and ranking while experimenting.

4.6.1 *Method 1:*

The first method is very simple and naive. We try to find out the job postings that simply match with job title query. We search for the job title in all the different fields like job title, qualification, experience etc. The method did not prove very effective.

4.6.2 *Method 2:*

We envisioned to leverage the experience oriented keyword set for each job postings or resume in our retrieval system. As explained in the section 4.3.3 and section 4.3.4 we calculated these during the document analyzer part. Next we obtain the experience oriented keyword set from the query itself. With all these in hand, we then try to identify which document nearly match with the generated keyword set. If the document match with more than 70 percentage of the keyword set, we boost the document scores. Retrieval scores were calculated using the TF-IDF method. Other scoring algorithms (i.e. BM25) could be used in our system easily.

4.7 User Interface Design

We have built a prototype to implement each major component of our system. The front end was built with HTML and CSS while Java Server Page facilitated the client connection. Finally, we built the back-end using Apache Tomcat Server 6.0 and lucene. A screenshot of the front-end is displayed in figure 4 and 5.

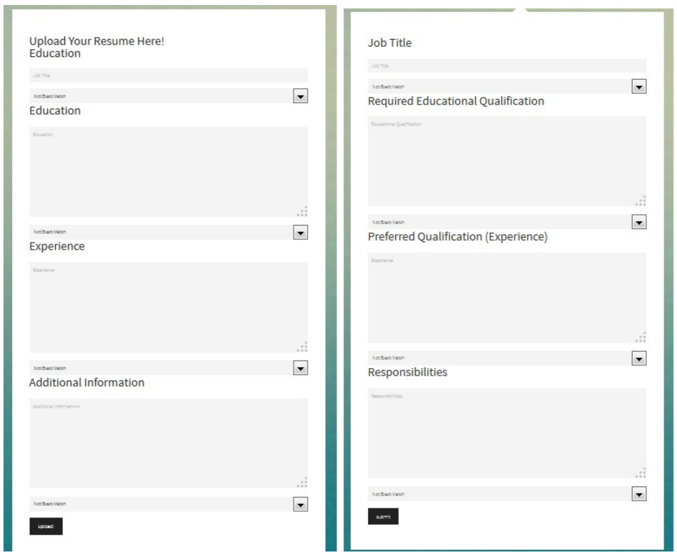
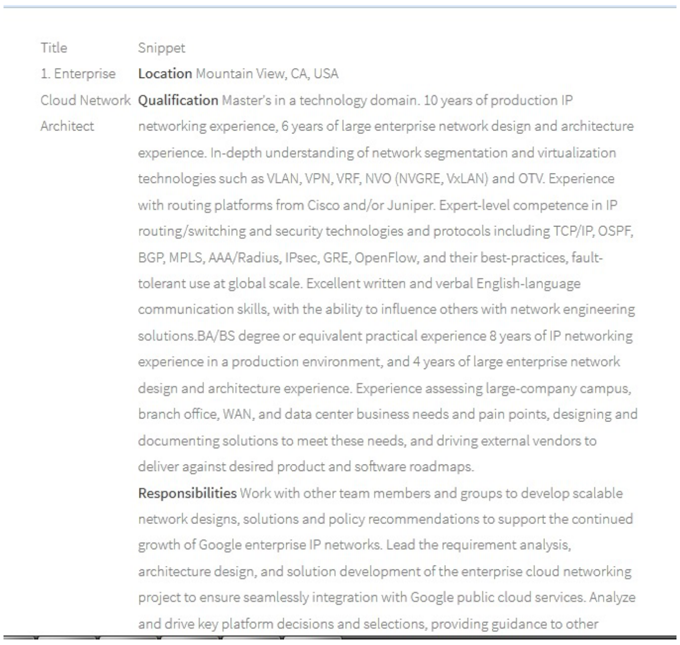


Figure 4: A sample input interface of our system



Title | Snippet

1. Enterprise Cloud Network Architect

Location Mountain View, CA, USA

Qualification Master's in a technology domain. 10 years of production IP networking experience, 6 years of large enterprise network design and architecture experience. In-depth understanding of network segmentation and virtualization technologies such as VLAN, VPN, VRF, NVO (NVGRE, VxLAN) and OTV. Experience with routing platforms from Cisco and/or Juniper. Expert-level competence in IP routing/switching and security technologies and protocols including TCP/IP, OSPF, BGP, MPLS, AAA/Radius, IPsec, GRE, OpenFlow, and their best-practices, fault-tolerant use at global scale. Excellent written and verbal English-language communication skills, with the ability to influence others with network engineering solutions.BA/BS degree or equivalent practical experience 8 years of IP networking experience in a production environment, and 4 years of large enterprise network design and architecture experience. Experience assessing large-company campus, branch office, WAN, and data center business needs and pain points, designing and documenting solutions to meet these needs, and driving external vendors to deliver against desired product and software roadmaps.

Responsibilities Work with other team members and groups to develop scalable network designs, solutions and policy recommendations to support the continued growth of Google enterprise IP networks. Lead the requirement analysis, architecture design, and solution development of the enterprise cloud networking project to ensure seamlessly integration with Google public cloud services. Analyze and drive key platform decisions and selections, providing guidance to other

Figure 5: A sample result interface of our system

5. EVALUATION

In this section, we have to tried to evaluate our personalized job/resume recommendation system. As a baseline, we have used typical keyword based (job title, location, company) search engine. To analyze the performance we have decided to use the Mean Average precision (MAP).

It is to be noted that, calculation of MAP is fully dependent on the relevance label. So before performing any kind of performance analysis, we need to find out the following two things:

1. For a given resume, we have to find out a set of relevant job postings.

2. For a given job postings, we have to find out a set of relevant resumes.

Obviously, manually labelling around 6000 jobs for about 1450 resumes is really around impossible. So we carefully select 2 resumes form the database and select 10 relevant job postings related to those resumes. Then we mixed up these 10 job postings with more 20 more random job postings. Then we perform job retrieval using our system and traditional keyword based search engine. So for the manually labelled data we found that MAP of our system is 0.3395 where as the MAP of the traditional system is 0.295.

Due to absence of a large volume of relevant documents performing extensive experiment is quite impossible. Once we have these data we are thinking of calculating the effects the major component of our system. We will try to address the following questions:

1. Is automatic keywords identification is alone sufficient to provide better result?

2. What is the impact of adding the experience years with the keyword set?

3. What kind of document boosters heuristic perform better?

4. Is there any correlation between the document title and the automatic keyword set?

6. SOME ADDITIONAL FEATURES

There are some other notable features of our system which we are going to discuss in this section. The first one is the user control over field. If you notice our interface in figure 4, you will find that for each field there is option for exact match or not. If user chooses the exact match option then the term must have to appear in the document else the condition can be relaxed.

The next important feature of our system that it also help user to find suitable job company. To provide this we leverage some new insight. Rather than simply finding the relevant job postings this time we also find similar resume. That means we are performing here resume search on the resume index. Then we select a list of companies from these top relevant resume. Our underlying assumption here is that many of the resume holders are working as an employee in some companies.

7. DATA ANALYTIC

After crawling Facebook, Google, and IBM job postings for just over a month we collected information on over 6,000 jobs. In order to perform analytics on the job postings we inserted the data into a SQL database. One key characteristic for each of these jobs is the date the job was posted. While running our crawler, if we found a new job posting, we would mark the date as first seen. If the job we are crawling match a job we had crawled previously, we would keep the date first seen the same, but update the date last seen to the current date. This method meant that jobs that were taken down would have a last seen date equal to the last time we saw the job posting while crawling. We could then use the difference between these two dates to find out exactly how long the job was listed on the companys' website.

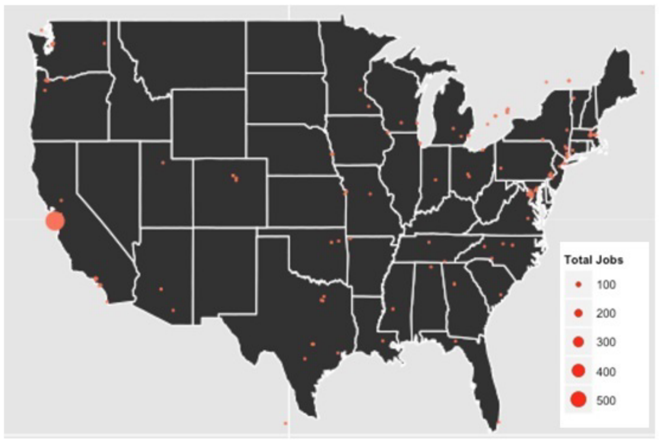


Figure 6: Map of Job Postings in United States

We ran this crawler every day for just over a month and were able to see how fast jobs were being filled and how fast companies were posting jobs. Of the 424 Facebook jobs, 32.5% were taken down after a month. Of the 3,939 IBM jobs, only 6.3% were taken down after a month. Although more jobs were being filled at IBM, the percentage of jobs at Facebook was higher. This could indicate that Facebook jobs are more highly sought after. However, since we only crawled the data during the month of November, we cannot accurately predict hiring behavior of these companies. If we continued crawling for an extended period of time, we could analyze trends for months. Ideally, we could identify when companies perform bulk hiring and therefore, we could inform potential job candidates when is the best time to apply. If Facebook hires four times as many people in September than any other month, then submitting job application before that time would be beneficial to the applicant. Hence these information are very useful for job applicants.

We also performed analytics on job locations. Figure 6 shows where the jobs we crawled are located and a total count at each location. In figure 6 the large red dot on the left side of the graphs is located in the San Francisco Bay Area, California. Google's headquarters, Mountain View, and Facebook's headquarters, Menlo Park, are both located near San Francisco. By mapping the location of job postings, we can provide a user insight to what companies are hiring near where they live. In addition, we can show a user where else a company is hiring. In case a job posting they want

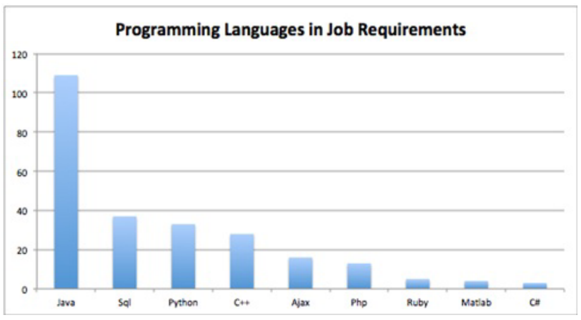


Figure 7: A count of job postings that require different programming languages

to apply for is not where an applicant wants to work, they could contact the company and request to work at a more convenient location.

Another point of interest for analytics stems from the job requirement section of each job posting. Figure 7 shows a count of how many job postings require different languages. In this graph, we narrowed down out data set to only include jobs in the "Software Development" and "Software Engineer" departments. We found the most popular programming languages these departments required were Java, SQL, Python, and C++. We compared our top five to a list of the most popular languages and found that C sharp was ranked 4th and SQL was ranked 9th. By analyzing what companies are currently looking for in applicants, we can provide insight to applicants. For example, if a student, who knows Java, is unsure which new programming language class to take, we might suggest learning SQL over C sharp, since SQL is a more marketable tool.

8. CONCLUSION

In this project, we have built a personalized data matching system to improve economic engagement by helping candidates and recruiters find qualified matches using an improved information retrieval system that processes full-text inputs. In current job-matching systems such as LinkedIn, Glassdoor, and Indeed.com, information retrieval systems are limited to simple key-word query searches including "job title", "keyword" or "company name". In some cases, candidates are only identified by a collection of "tags" where all semantic qualifiers are lost. Our system allows users to search for a job using their entire skill set, background, and experiences; likewise, employers can search for candidates using large paragraphs of texts with rich qualifying information that improves search specificity. Preliminary, but continued tests indicate that our system outperforms the current model in Mean Average Precision, but more extensive feedback testing will be needed to determine significance. Outside of these retrieval metrics, our system fundamentally improves the way we search for jobs by comprehensively tailoring query search to our personal history and skill level.

As our database grows, we will be able to build a robust graph of the economy across job types, industries, and candidates. This graph will enable us to perform important analytics which we can use to enhance the user's experience to give them a more holistic perspective of the economy. We hope our toolset can be used to help students, graduates, and companies engage in the global workforce in ways best utilize their unique talents and abilities.

9. FUTURE WORK

As a future work, we will continue to build and grow our system, we would like to provide users with specific feedback tailored to their job-seeking needs. For example, if a software engineer is seeking a job from a specific company or within a certain field, can we provide them feedback on the most common skills required for that role? If their minimum requirements match a majority of jobs they are looking for, can we recommend additional skills or experiences for the job? If they are missing key requirements, what is the average, baseline skillset needed for the work they seek? These recommendations will be tailored specifically to each individual given the personalized nature of our recommendation engine. Additionally, we can perform resume-to-resume searches to identify patterns among users and recommend skillsets or contact information based on matching data within our resume index.

Second, we would like to continue to expand our "economic graph" database of job postings and resume submissions. As we grow our database over time, we will have a thorough and comprehensive sense for the kinds of jobs that are offered in different parts of the world. In this sense, our analytics will not only tell us something about the specific qualifications and requirements of jobs that are immediately available, they will also reveal patterns about the economy of various industries over time. This insight could be invaluable. In order to expand our crawling proficiency, we could use MozendaTM or another web-content extractor to streamline the data-extraction process. Moreover, we plan on submitting our prototype to the LinkedIn Economic Graph Challenge which seeks to create greater economic opportunity and engagement for the 3 billion people in the global workforce [2]. If our submission is selected among a pool of winners, we will have access to LinkedIn's robust database to improve our IR system and create better predictive models.

Finally, we are excited to expand our job-analytics component of the project to discern patterns in the economy. What time of year do certain industries recruit heavily for new talent? How long do coveted jobs postings last on a company's website? As we expand both our database and our analytics, we will be able to provide constant and personalized feedback to improve the job-searching process. For example, if we know a user has a certain skill level and interest, can we alert him about new jobs that fit his profile and patterns? Perhaps we can even predict how many weeks he has to apply for a job before the post is filled! Finally, if we can overlay data about their social graph or LinkedIn connections, we can suggest a "networking trail" to the job of their dream.

10. REFERENCES

[1] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[2] LinkedIn. Economic graph challenge. November 2014.

[3] Y. Lu, S. El Helou, and D. Gillet. A recommender system for job seeking and recruiting website. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 963–966. International World Wide Web Conferences Steering Committee, 2013.

[4] R. Rafter, K. Bradley, and B. Smyth. Automated collaborative filtering applications for online recruitment services. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 363–368. Springer, 2000.