2017 IEEE Third International Conference on Big Data Computing Service and Applications

CaPaR: A Career Path Recommendation Framework

Bharat Patel, Varun Kakuste, Magdalini Eirinaki
Computer Engineering Department
San Jose State University
San Jose, CA, USA

***Abstract*—In today's world, recommendation systems are used to solve the problem of information overload in many areas allowing users to focus on important information based on their interests. One of the areas where such systems can play a major role is in helping students achieve their career goals by generating personalized job and skill recommendations. At present, there are many job posting websites providing a huge amount of information and students need to spend hours to find jobs that match their interests. At the same time, existing job recommendation systems only consider the user's field of interest, but do not take into consideration the user's profile and skills, which can generate more relevant career recommendations for users. In this work, we propose CaPaR, a Career Path Recommendation framework, which addresses such shortcomings. Using text mining and collaborative filtering techniques the system first scans the user's profile and resume, identifies the key skills of the candidate and generates personalized job recommendations. Moreover, the system recommends additional skills to students required for related job openings, as well as learning resources for each skill. In this way, the system not only allows its users to explore large amounts of information, but also expand their portfolio and resume to be able to advance their careers further. We experiment and evaluate the various recommendation algorithms with real-world data collected from the San Jose State University career center web site.**

Keywords—recommendation systems, NLP, collaborative filtering, job recommendations, skill Recommendations, career path

I. Introduction

Due to the rapid developments in Internet technology, more and more job seekers have started sharing their academic and professional information on the Web. At the same time, enterprises started leveraging job posting sites for recruiting. According to Jobvite's report, 68% of online jobseekers are college graduates or post graduates and 94% of recruiters use, or plan to use social media for recruiting [1]. This number has increased steadily for the last 6 years. Moreover, 30% of all Google searches, about 300 million per month, are employment-related.

As information continues to grow, searching for a job online often becomes an overwhelming and tedious activity, especially for students and young professionals who still work on building their resume. Even when students don't have to go to more generic web sites and resort to their University's career site, such sites lack the "intelligence" of matching resumes to jobs, other than simple filtering by major. What is more important, is an opportunity for such sites to serve also as career path advisors, identifying emerging trends and needs of the job market, and highlighting desired skillsets to the job seekers.

In this work we try to address the aforementioned problems, by presenting CaPaR, a Career Path Recommendation framework focusing on students and young professionals. CaPaR employs text mining capabilities, and combines them with collaborative filtering algorithms, to perform two different types of recommendations to its users, namely job and skill recommendations, such that the users can both find a job, but also identify areas and skills that are currently desired in the job market, but they might still be lacking, such that they can work on attaining them.

The rest of this paper is organized as follows: in Section II we provide a brief overview of the related work and state-of-the-art in the area of job search and personalized job recommendations. We provide a high-level overview of the system in Section III and discuss in more detail the text mining and recommendation engine modules in Sections IV and V respectively. A brief overview of the CaPaR prototype is given in Section VI, while we discuss in detail the experimental evaluation of the system in Section VII. Finally, we conclude with our plans for future work in Section VIII.

II. Related Work

A. Literature Survey

Several researchers have looked into the problem of career path recommendations. The majority is focusing on job recommendations, while only a handful look into career path recommendations. For example, Paparrizos et al. trained a machine learning model to predict candidates' next job transition based on their past job histories as well as the data of both candidates and enterprises in the web [2]. Some papers employ metadata and external knowledge to make recommendations. Chien et al. developed a data mining framework based on decision tree and association rules to generate useful rules for selecting personnel features, such as work experience and job performance [3]. Most of the research approaches employ content-based, collaborative filtering, or a hybrid approach to generate job recommendations [4]. Almalis et al. propose a content-based recommendation algorithm which extends and updates the Minkowski distance in order to address the challenge of matching people and jobs [5]. The proposed algorithm uses a structured form of the job and the candidate's

978-1-5090-6318-5/17 $31.00 © 2017 IEEE
DOI 10.1109/BigDataService.2017.31

profile, produced from a content analysis of the unstructured form of the job description and the candidate's CV. Zhang et al. compare user-based and item-based recommendation algorithms incorporating information from user resumes to generate job recommendations [6]. Fazel-Zarandi and Fox combined different matchmaking strategies in a hybrid approach for matching job applicants and jobs by using logic-based and similarity-based matching [7]. Lu et al. proposes a hybrid system incorporating a PageRank-like algorithm with content-based recommendations [8]. A slightly different line of work is employing user clustering to enhance the recommendation process [9, 10]. Finally, Bakar and Ting proposed a Bayesian network-based solution for recommending soft skills required for each job, using a dataset collected via extracting information from advertisements and also through interview sessions with a few identified experts [11]. To the best of our knowledge, this is the best effort to propose an integrated job and skill recommendation framework focusing on students and young professionals.

B. *State of the Art*

One category of applications currently available to help users in achieving their career goals is job recommendation systems. We present here the most popular job recommendation systems currently available.

LinkedIn[1] is a professional social network that also acts as a job posting and job search portal. LinkedIn recommends jobs to users based on content-based and collaborative filtering recommendation, using similar users as represented by their jobs as input [12]. The current version of LinkedIn does not provide skill recommendations to users based on their job interests. Recently, the company launched LinkedIn Students, an application that addresses specifically the current student population and employees interested in recruiting from this market. In its current form, they built upon their existing data and infrastructure, providing a “network within the network” environment for students. Our framework can act complementary to this application, especially the skill recommendation component.

Glassdoor[2] is a job posting and job search portal. Glassdoor recommends jobs to its users based on content-based recommendation strategy where recommendations are generated based on (similar) jobs the user has previously browsed [13]. Glassdoor does not provide capabilities of skill recommendations to achieve particular job and course recommendation to students.

Angellist[3] is also a job posting and job search application. However, it is limited to small scale and startup level companies. It recommends jobs to users based solely on their preferences. AngelList does not provide capabilities of personalized job recommendations or skill and course recommendations to users.

III. System Architecture

The high-level architecture of the CaPaR framework is shown in Figure 1.

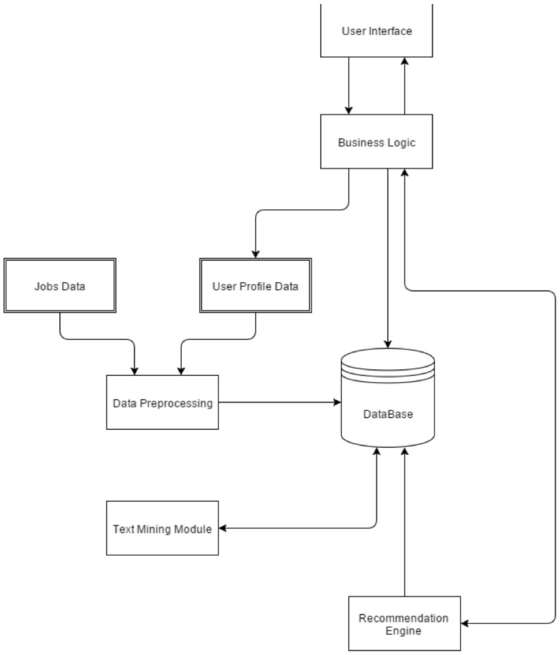


Figure 1. System Architecture Diagram

Each user needs to create an account by interacting with the User Interface module. Through this module, the user logs-in and creates their professional profile by uploading their resume and setting some preferences (e.g. job location etc.). The Business Logic module processes requests sent by user while interacting with system. The users’ profiles are processed and stored into the database using Data Preprocessing module. A similar preprocessing is being applied to the job announcement data. The Text Mining module fetches the user's profile data including the educational and the job description details from the database (NoSQL) and then outputs the required data needed as input to the Recommendation Engines. The Recommendation Engine module consists of various sub-engines, backed up by content-based, collaborative filtering and hybrid recommendations. These modules take as input the user's profile, preferences, and current skills. Depending on the engine, the output differs, resulting in two types of recommendations, namely job and skill recommendations., aiming at generating various types of personalized recommendations. In what follows, we provide a more detailed overview of the Text

[1] https://www.linkedin.com/

[2] https://www.glassdoor.com/index.htm

[3] https://angel.co/

Mining module in Section IV and the various Recommendation Engine submodules in Section V.

IV. Text Mining Module

The Text Mining module performs two distinct tasks, namely parses the user-provided resumes as well as the employee-provided job descriptions, to create user and job profiles respectively.

The input to the resume parsing submodule is the user's resume, uploaded as part of the user's profile creation. The resume is parsed and each word in it is considered as a token. The parsed token is compared against a thesaurus, containing a predefined set of skills. After parsing the skills from the user's resume, the system asks them to weight their proficiency in each of the skills either as Beginner (1), Intermediate (2) or Expert (3) as a part of profile creations step (a snapshot of this process is shown in Figure 6). The output of this process is a set of <user, skill, weight> triplets, with each user being uniquely identified by a user-id. These triplets form the user profiles used as input to the recommendation process. So for instance, for user A, who is proficient in Java and beginner in Python, the user profile will be the set of the two triplets <A, Java, 3> and <A, Python, 1>.

The job parsing process is very similar, yet slightly more complicated. The system fetches job announcements posted in the host site (in our system implementation we used San Jose State University's Career Center as the host job posting website) and for each job it extracts some basic information (e.g. company name, job title, etc.) as well as the required skills, using the same skill thesaurus that was used in the resume parsing module. If the job description provides different levels of skills (e.g. required vs. desired vs. good to have), the system assigns the respective weights (3, 2, and 1, respectively). The output of this process is a content-rich job profile, as shown in Figure 2. This profile consists of content-based features, as well as skill triplets <job, skill, weight>, similar to the ones created for the users.

V. Recommendation Engines

The core functionality of the CaPaR system is to provide personalized job and skill recommendations to its users. The entire process of the Skill Recommendation submodule is shown in Figure 3, while the architecture of the Job Recommendation submodule is shown in Figure 4.

A. Skill Recommendation Engine

Using the user profile triplets derived from their resumes, we can represent each user as a vector $u = <s_1, ... s_m>$, where s_i represents a distinct skill and m is the total number of distinct skills included in the thesaurus. These vectors are weighted, with each element taking values from 0 to 3, depending on the user's proficiency of that skill (3 being the highest), and 0 representing absence of this skill.

The existing user profiles form a n x m user-skill matrix (where n is the total number of users in our database). This matrix is the input to the user-based collaborative filtering process as follows. We first calculate all user-based similarities *sim(u,v)* using the Pearson correlation coefficient:

$$sim(u,v) = \frac{\sum_{i\epsilon C}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i\epsilon C}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i\epsilon C}(r_{v,i} - \bar{r}_v)^2}}$$

where $r_{u,i}$ represents the level of proficiency of user u for skill i, and C represents the set of skills that both users u and v have.

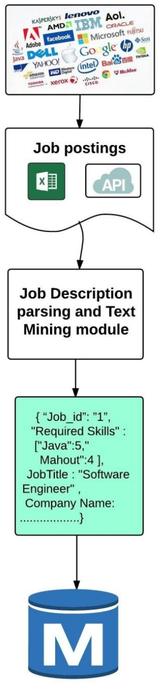


Figure 2. Jobs Parsing Module

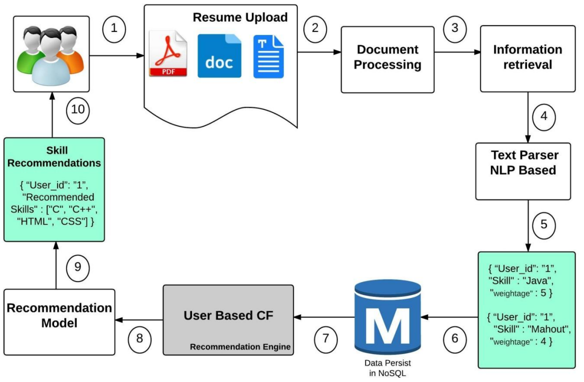


Figure 3. Skill Recommendation Process

Once the similarities are calculated, the algorithm predicts the preference scores of all missing skills based on the intuition that similar users will have similar skills, in similar levels of proficiency. This is achieved by calculating the weighted average of skill proficiency of all similar users $d \in D$for each missing skill s of the current user u:

$$r_{u,s} = \frac{\sum_{d \epsilon D} sim(u,d)\, r_{d,s}}{\sum_{d \epsilon D} sim(u,d)}$$

where *sim(u, d)* denotes the similarity between users *u* and *d.* The size of the "neighborhood" of user *u,* i.e. the number of similar users included in set *D* is defined via cross-validation.

For example, if user A is excellent in Java and beginner in Python, and (similar) user B is excellent in Java and R and beginner in Python, the system will recommend R as a new skill to user A.

B. Job Recommendation Engine

In order to generate job recommendations, we designed a two-level hybrid recommendation engine, that combines content-based and collaborative filtering. The choice of a two-tier hybrid recommendation approach was made in order to avoid cold start and scalability problems [14].

For each user, the input to this recommendation engine is the user profile vector u = <s_1, ... s_m>, and the k x m job-skill matrix (where k represents the number of available job postings in the database), each row of which represents one job, as the vector j = <s_1, ... s_m>, where s_i represents a distinct skill and the level of importance of this skill for the particular job (similarly to user profile u, the weights of these elements range from 1 to 3, where 1 means that the user is beginner, 2 is intermediate and 3 means expert in that skill).

Job Recommendations

In the first level of recommendations, the engine employs content-based filtering in order to find the most similar jobs to the user's profile. To achieve this, the system calculates all similarities between the current user's profile vector u and all jobs $j \in J$, using the K-NN algorithm and Euclidean distance between the vectors (size of neighborhood N is defined experimentally via cross-validation, set to N=4 for the dataset we employed). This provides a fast way to find the most similar jobs in terms of skill overlap with the user's skills. However, since this first round returns very few results, we expand the number of jobs returned as recommendations, in the second level.

The second level of the hybrid recommendation engine takes as input the neighborhood SJ_u including the jobs that were output by the first level of the engine. Using these as input, the algorithm employs item-based collaborative filtering among all jobs in the database, to retrieve other jobs that are similar to the initial seed.

The output of this process is a set of jobs that are ranked more similar to the user's profile. We need to point out that the algorithm improves as it collects more data from the user. For instance, as soon as a user selects a recommended job to apply this job will be used to enhance the user's profile. For example, if user applies to any of the recommended jobs, that job will be considered as user's preferred type of job. So during next run of recommendation algorithm jobs similar to previously applied jobs will also be provided in the list of recommended jobs. Therefore, each time the user will receive updated recommendations.

Job-related Skill Recommendations

The system also recommends skills to the user for each job they want to apply for, even if this job is not included in their recommended job list. This is achieved through a simple binary comparison of the user's skill set to the job's required skills. After the comparison, the system recommends skills which are required for that particular job and are not available in user's current skill profile. A snapshot of the outcome of this process is shown in Figure 7.

Jobs recommendation for New Users (Cold Start Problem)

The recommendation engine runs periodically, updating the job-skill and user-skill matrices and the respective similarity metrics and recommendations. However, when a new user registers in the system and creates a profile, the system cannot provide recommendations until the engine refreshes the matrices and respective similarities. In order to still be able to provide recommendations, even to new users, immediately after they register and create their preference profile, we follow a slightly different process, as shown in Figure 5. More specifically, we follow a content-based approach, to rank and recommend the jobs that are the most similar to the top-5 (in terms of proficiency) user skills. We should note that this set of recommendations is temporary, and is replaced by the ones generated by the hybrid recommendation engine, as soon as the system runs again.

VI. CAPAR PROTOTYPE

Using real data from San Jose State University's Career Center, including job postings aimed at Computer Science and Engineering majors, and real resumes of students of the Computer Engineering department, we developed a proof-of-concept system prototype. Figures 5 - 7 show some snapshots of the system's prototype, during the various steps of the process (resume parsing, skill recommendations, job recommendations). For the implementation, we employed various tools from the Hadoop ecosystem, namely the Apache Tika parser library for the Text Mining module, the Apache Mahout recommendation algorithm libraries for the Recommendation Engines module, and the NoSQL MongoDB database.

The recommendation engine runs every half an hour and generates new jobs and skills recommendation for users. The algorithm considers profiles of new user generated after last run and also considers already existing users' profiles. So it updates the recommendations for existing users and generates new recommendations for newly registered users every 30 minutes.

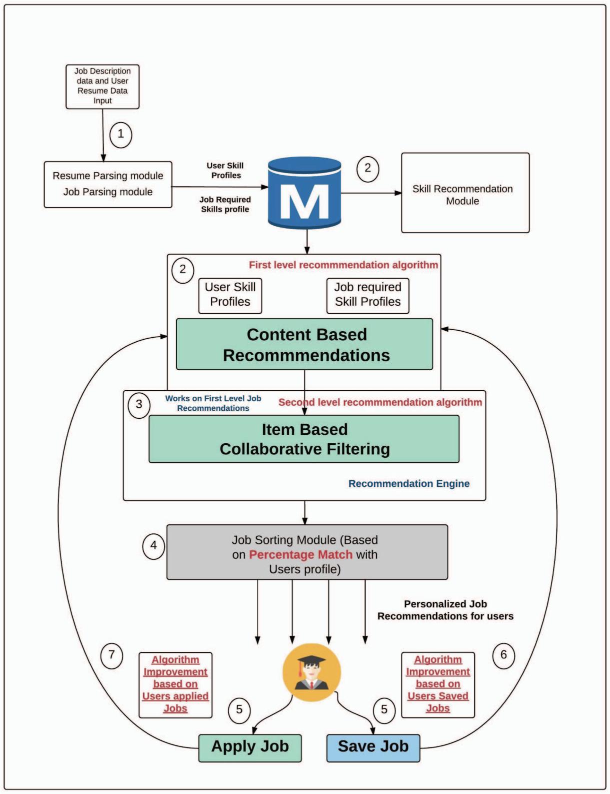

Job Description data and User Resume Data Input
1
Resume Parsing module
Job Parsing module
User Skill Profiles
Job Required Skills profile
M
2
Skill Recommendation Module
2
First level recommmendation algorithm
User Skill Profiles
Job required Skill Profiles
Content Based Recommmendations
3
Works on First Level Job Recommendations
Second level recommmendation algorithm
Item Based Collaborative Filtering
Recommendation Engine
4
Job Sorting Module (Based on Percentage Match with Users profile)
Personalized Job Recommendations for users
7
Algorithm Improvement based on Users applied Jobs
6
Algorithm Improvement based on Users Saved Jobs
5
5
Apply Job
Save Job

Figure 4. System Architecture Diagram

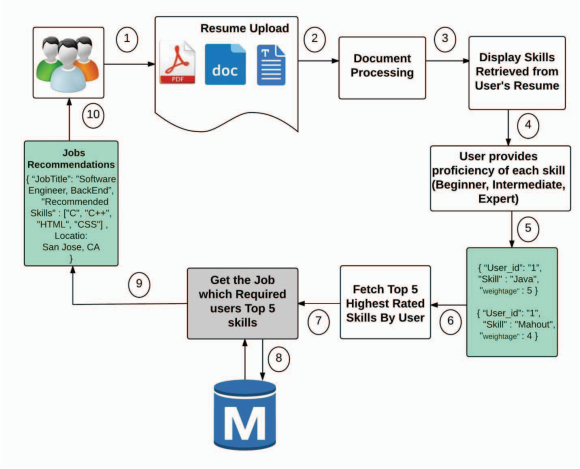
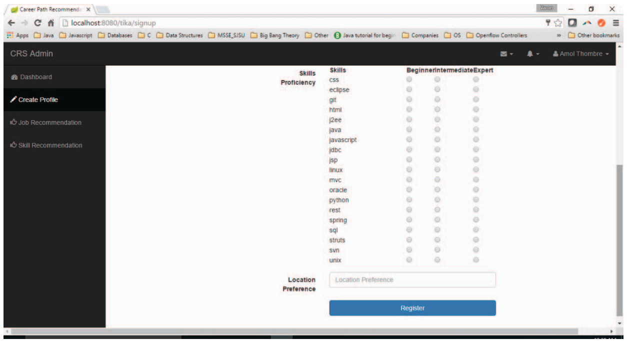

1
Resume Upload
PDF
doc
2
Document Processing
3
Display Skills Retrieved from User's Resume
4
User provides proficiency of each skill (Beginner, Intermediate, Expert)
5
{ "User_id": "1", "Skill" : "Java", "weightage" : 5 }
{ "User_id": "1", "Skill" : "Mahout", "weightage" : 4 }
6
Fetch Top 5 Highest Rated Skills By User
7
Get the Job which Required users Top 5 skills
8
M
9
Jobs Recommendations
{ "JobTitle": "Software Engineer, BackEnd", "Recommended Skills" : ["C", "C++", "HTML", "CSS"] , Locatio: San Jose, CA }
10

Figure 5. Job Recommendations for New User (Cold Start)


Career Path Recommenda
localhost:8080/tika/signup
Apps Java Javascript Databases C Data Structures MSSE_SJSU Big Bang Theory Other Java tutorial for begin Companies OS Openflow Controllers Other bookmarks
CRS Admin
Amol Thombre
Dashboard
Create Profile
Job Recommendation
Skill Recommendation

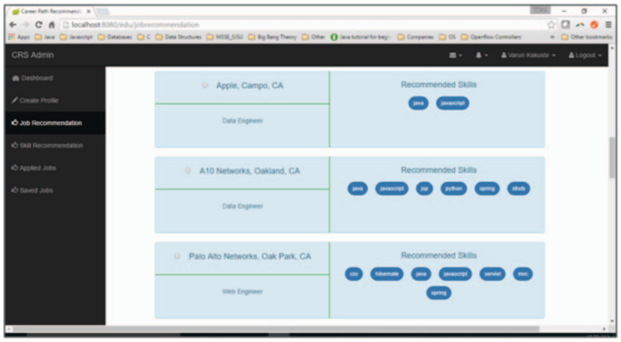
Skills Proficiency

Skills	Beginner	Intermediate	Expert
css	○	○	○
eclipse	○	○	○
git	○	○	○
html	○	○	○
j2ee	○	○	○
java	○	○	○
javascript	○	○	○
jdbc	○	○	○
jsp	○	○	○
linux	○	○	○
mvc	○	○	○
oracle	○	○	○
python	○	○	○
rest	○	○	○
spring	○	○	○
sql	○	○	○
struts	○	○	○
svn	○	○	○
unix	○	○	○

Location Preference: Location Preference

Register

Figure 6. Skills Parsed from Resume


Career Path Recommend
localhost:8080/edu/jobrecommendation
Apps Java Javascript Databases C Data Structures MSSE_SJSU Big Bang Theory Other Java tutorial for beg Companies OS Openflow Controllers Other bookmarks
CRS Admin
Varun Kakuste
Logout
Dashboard
Create Profile
Job Recommendation
Skill Recommendation
Applied Jobs
Saved Jobs

Job	Recommended Skills
○ Apple, Campo, CA Data Engineer	java javascript
○ A10 Networks, Oakland, CA Data Engineer	java javascript jsp python spring struts
○ Palo Alto Networks, Oak Park, CA Web Engineer	css hibernate java javascript servlet mvc spring

Figure 7. Job Recommendations for User

VII. EXPERIMENTAL EVALUATION

A. Dataset

The job description dataset was provided by San Jose State University's career center. The data was neutralized and included jobs posted by companies on SJSU career center's portal earlier in the year, focusing on Computer Science and Engineering-related postings. For the experiments, we split the data into training data and test data on a 70:30 ratio. We also collected user resumes from SJSU students of the Computer Engineering department. The entire dataset consists of 10,000 job descriptions and 100 users. Some results discussed below are based on 30 users and 3000 job descriptions. All the algorithms are implemented using Apache Mahout machine learning library.

A. Evaluation of skills recommendations algorithm (user-based CF)

Similarity metric

Since the skill recommendation algorithm makes predictions on the weights of skills the user does not have, we employed two metrics to evaluate its performance: precision and RMSE (Root Mean Square Error). Precision measures how well the algorithm predicts that a user needs to have a specific skill, while RMSE measures how well the algorithm predicts the level of proficiency of this skill. We compared three variations of the user-based CF algorithm, depending on the similarity metric used. As shown in Table I, the Pearson Correlation coefficient variation gives both the lowest RMSE and highest precision. Therefore we adopt this variation for our system prototype.

TABLE I. Skills Recommendation Algorithm Evaluation of Similarity Metrics

Recommendation algorithm technique	Evaluation Method	Similarity Measure		
		Euclidean distance	Pearson Correlation	Log Likelihood
User-based CF	RMSE	1.1081	0.9380	1.1456
	Precision	0.7389	0.7989	0.4530

Neighborhood Size

Figure 8 shows the RMSE values for the three variations of the algorithm, when we change the size of the nearest neighborhood (i.e. how many similar users are considered for making predictions and generating recommendations). From this we conclude that Pearson correlation provides the best solution with lowest RMSE values for N equals to 1, 2, and 3. Given these findings, we adopted N=2 for our system prototype.

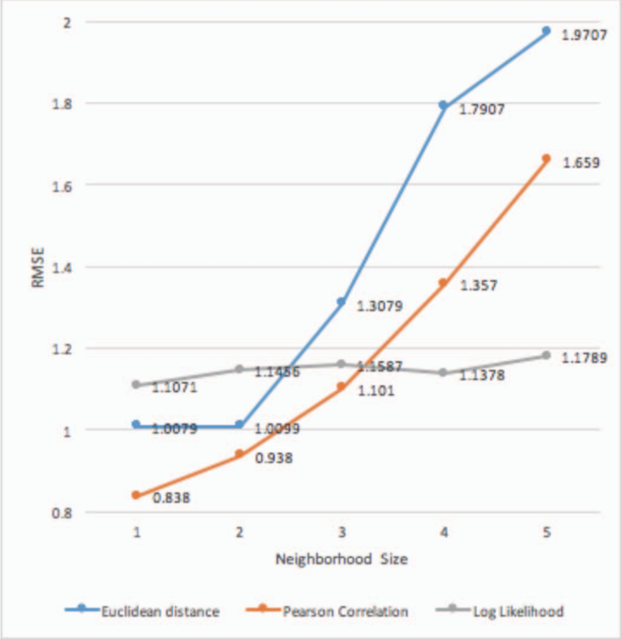


Figure 8. Nearest Neighborhood Size Comparison for Skills Recommendation Algorithm

Accuracy of Skills Recommendation algorithm

In order to calculate the accuracy of the skills recommendation algorithm we perform the following experiment: we randomly selected 30 users and for each user we hid 40% of their already available skills and apply the algorithm to the updated profile. We then calculate the recall of the algorithm, defined as: $r = \frac{tp}{tp+fn}$ where *tp (true positives)* are the predicted skills that were included in the user's original profile, and *fn (false negatives)* are the skills that were included in the user's original profile but not predicted by the algorithm.

For example, user A has skills {C, Java, C++, Apache Mahout, Spark, Python, Tableau, AWS, HTML, CSS}. The algorithm will hide 40% skills, for example C++, HTML, Python and CSS. The updated user profile will be{C, JAVA, Apache Mahout, Spark, Tableau, AWS}. Let's assume that the skills recommendation algorithm recommends three skills: C++, HTML and CSS to user A. In this case, recall is ¾, so we conclude that the accuracy of algorithm is 75%. The average recall for the 30 users following was 65%.

B. Evaluation of jobs recommendations algorithm (Hybrid recommendation algorithm)

As discussed previously, we employed a two-level hybrid recommendation algorithm which is combination of content-based and item-based CF. In our experiments we evaluated both algorithms (separately) to select the best parameters for each level of the hybrid recommendation algorithm.

Content-based Recommendations

Similar to the skill recommendation algorithm, we employed RMSE and precision to evaluate the performance of three variations of the content-based algorithm, depending on the similarity metric used. As shown in Table II, the Euclidean distance gives both the lowest RMSE and highest precision. Therefore we adopt this variation for our system prototype.

TABLE II. Evaluation for Job Recommendation Algorithm (Level One)

Recommendation algorithm technique	Evaluation Method	Similarity Measure		
		Euclidean distance	Pearson Correlation	Log Likelihood
Content-based	RMSE	1.0220	1.1207	1.1456
	Precision	0.7387	0.6128	0.5529

Moreover, we evaluated the algorithm for various neighborhood sizes. Figure 9 provides the RMSE comparison between the three algorithm variations, and for different values of N. We observe that the euclidean distance variation provides the best results for N=4, and this is what we adopt for our prototype.

Item-based CF algorithm

As explained previously, the outcome of the first level of recommendation is a set of jobs whose skills match best those of the user. This output is used as input to the second level of the recommendation engine, that implements an item-based collaborative filtering algorithm. Using this approach, the original seed set of jobs is further expanded with additional jobs that have similar skill requirements.

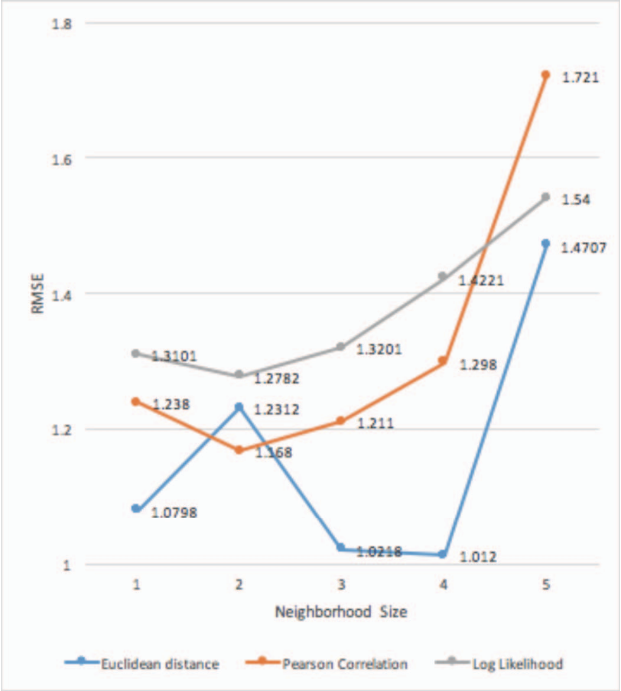


Figure 9. Nearest User Comparison for Job Recommendation (Level One)

We again evaluated three variations of the item-based CF algorithm with three different similarity measures, namely Euclidean distance, Pearson correlation, and log likelihood. As shown in Table III, the best results are provided by the Pearson correlation variation, which provides the lowest RMSE and highest precision values. We should point out that for the item-based collaborative filtering, we did not perform a neighborhood evaluation, as this is not a parameter of the Apache Mahout implementation.

TABLE III. Evaluation for Job Recommendation Algorithm (Level Two)

Recommendation algorithm technique	**Evaluation Method**	**Similarity Measure**		
		Euclidean distance	**Pearson Correlation**	**Log Likelihood**
Item Based CF	RMSE	1.3518	1.2490	1.4021
	Precision	0.5978	0.6045	0.5903

Table IV summarizes the optimal parameters and similarity metrics for the two submodules of CaPaR's recommendation engine, selected after the experimental evaluation. These are the default values used in CaPaR's prototype.

TABLE IV. Default parameters for Recommendation Engine

	Recom-mendation algorithm	**Similarity Measure**	**Nearest Neighbor hood**	**Input Data**
First Level	Content Based Recommendation (Modified Mahout User-Based CF)	Euclidean Distance	4	User Profile and Job Requirement in the form of Job and Weightage
Second Level	Item Based Collaborative Filtering	Pearson Correlation	N/A in Item based CF Mahout algorithm	Job Recommendations generated in First Level of recommendation

VIII. Scalability

In this paper we presented the evaluation's with the dataset consists of 10,000 job descriptions and 100 users. The application architecture is designed to scale as required. The system can run on cloud infrastructure using Apache Spark or Hadoop cluster which can handle more number of job descriptions and more number of users. To increase the system capacity at any time. new machines can be added in the existing cluster without affecting the running application.

IX. Conclusions and Future Work

In this paper we presented novel career path framework for personalized job and skills recommendations, focusing on students and young professionals. We presented the architecture and main components of the framework, as well as a system prototype implementation, employing real data from the San Jose State University career center and students. We also provided an experimental evaluation of the two recommendation engines that form the core of this framework.

In the next phase of this project we plan to focus on incorporating details of courses taken by the students during their studies, and also alumni data (on skills and acquired job experience) to enhance the input dataset and make the experience even more personalized. Another direction is to crawl the Web and retrieve online courses and tutorials that address the recommended skills, such that the users can start training in order to enhance their resume.

IX. References

[1] S. Zheng, W. Hong, N. Zhang, F. Yang. "Job recommender systems: A survey", presented at 7th International Conference on Computer Science & Education (ICCSE), Melbourne, VIC, 2012

[2] I. Paparrizos, B. B. Cambazoglu and A. Gionis, "Machine learned job recommendation," In Proceedings of the fifth ACM Conference on Recommender Systems, pp. 325-328, Chicago, USA, October, 2011.

[3] C. F. Chien and L. F. Chen, “Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry,” Expert Systems with Applications, vol. 34(1), pp. 280-290, 2008

[4] W. M. Al-Adrousy, H. A. Ali, T. T. Hamza. “A Framework for Career-Education Hybrid Recommender System using a Selective Path Delta-SimRank Algorithm”. International Journal of Computer Application(0975-8887), Vol. 90, No. 2, March 2014

[5] N. D. Almalis, G. A. Tsihrintzis, N. Karagiannis and A. D. Strati, "FoDRA — A new content-based job recommendation algorithm for job seeking and recruiting," *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*, Corfu, 2015

[6] Y. Zhang, C. Yang, Z. Niu. “A Research of Job Recommendation System Based on Collaborative Filtering”, presented at Seventh International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2014

[7] M. Fazel-Zarandi and M. S. Fox, “Semantic matchmaking for job recruitment an ontolgy based hybrid approach,” In Proceedings of the 3rd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web at the 8th International Semantic Web Conference, Washington D. C., USA, 2010.

[8] Y. Lu, S. E. Helou, D. Gillet, “A Recommender System for Job Seeking and Recruiting Website”, WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.

[9] W. Hong, S. Zheng, H. Wang. “A Job Recommender System Based on User Clustering”. Journal of Computers, vol. 8, no. 8, August 2013

[10] Q. Nguyen, T. Huynh, T.N. Hoang, “Adaptive methods for job recommendation based on user clustering**”,** in Proceedings of the 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), 2016

[11] A. A. Bakar and Choo-Yee Ting, "Soft skills recommendation systems for IT jobs: A Bayesian network approach," *Data Mining and Optimization (DMO), 2011 3rd Conference on*, Putrajaya, 2011, pp. 82-87.

[12] A. Gupta and A. Aijaz. “Recommendations@LinkedIn,” presented at the Hadoop World Conference, Germany, 2011.

[13] R. Aylward. “Glassdoor Expands Its Integrated Social Platform With Neo4j” Internet, retrieved from http://info.neotechnology.com/rs/neotechnology/images/Glassdoor.pdf, October 2014

[14] P. Melville, R. J. Mooney, R. Nagarajan. “Content-Boosted Collaborative Filtering for Improved Recommendations”, in Proc. AAAI-02, 2002