Job Recommendation System based on Machine Learning and Data Mining Techniques using RESTful API and Android IDE

Harsh Jain
Software Developer,
App Street Software Pvt. Ltd
hiharshjain@gmail.com

Misha Kakkar
Department of Computer Science,
Amity University, Uttar Pradesh
mkakkar@amity.edu

***Abstract*— In the current Capitalist world with an abundance of different state-of-the-art industries and fields cropping up, ushering in an influx of jobs for motivated and talented professionals, it is not difficult to identify your field and to persevere to get a job in the respective field but lack of information and awareness render the task difficult. This problem is being tackled by Job Recommendation systems. But not every aspect from the wide spectrum of factors is incorporated in the existing systems. For the "Job Recommendation System - Vitae" machine learning and data mining techniques were applied to a RESTful Web Server application that bridges the gap between the Frontend (Android Application) and the Backend (MongoDB instance) using APIs. The data communicated through APIs is fed into the database and the Recommendation System uses that data to synthesize the results. To make the existing systems even more reliable, here efforts have been done to come up with the idea of a system that uses a wide variety of factors and is not only a one-way recommendation system.**

***Keywords*— Data mining, Machine Learning, Recommendation Systems, Content Based filtering, RESTful API, Android Application, Job Recommendations**

I. Introduction

FOR every individual in this world self-sustenance is the most vital aspect in his/her life. It is for self-sustenance that they go to school, go to college and get a job. Self-sustenance is deeply linked with employment. It is the existence and type of employment that decides the level of self-sustenance that an individual can reach in his/her life. It also defines how his/her life will be i.e. Are they financially independent? How economically strong are they? Can they afford to have a family? How big can their family be?

A job is looked upon as something which defines the character of a professional and the basis on which one is judged in the world. Getting a job is still considered a hectic and a daunting task. One of the biggest concerns for fresh graduates is getting a job, a way to sustain themselves in the world, pertaining to their degree and their skills. But seldom do they get the job they wanted since they could want anything.

In the age of information where all things, small and big, are available on the internet there is still no platform where contrasting people, ranging from a daily-wage laborer to a Chief X Officer (CXO) of an MNC, can look for jobs. Vitae (Job Recommendation System) aims to make that possible by providing one common platform for both aspirant employees/freshers and recruiters.

As far as the current job scenarios is concerned, the deserving candidates are losing out on jobs due to a varied spectrum of intrinsic and extrinsic reasons leading to a waste of Human Resources [3]. Job searching also poses as an exhaustive task at times as job seekers do not always know what, where and how to apply. Inability to get jobs even when a person has adequate skills creates a tense environment for both the individual and the people connected to him/her.

Also, there exists a problem of unregulated labor because of which many citizens of our country are at the mercy of their employers and are not treated as they should be and are denied humanitarian behavior. There is no fixed salary or project duration. This application also aims to transform the unregulated employment section of the country.

Online recruitment is already the major mode of recruitment done my major companies. It has already transformed the way companies recruit their employees [1][5]. 'Vitae' can further streamline this process right from the application process to the interview process for both the candidates and the companies. The candidates will know which companies to apply for and the companies would know which candidates to call for interviews without manual scrutiny.

II. Literature Survey

A Recommendation System, in simple language, is a classification and information filtering system which shows the user(s) material and information tailored around the profile, history and current data of the user(s). Recommendation Systems are used in almost every strand of the digital world.

"Suggested Videos" on YouTube, "People you may know" on Facebook, "Games, Books and Apps suggested for you" on Google Play Store etc. are all examples of Recommendation Systems/Engines which take into account the user's past watch/download history and suggest results which try to predict what the user might watch/download in the future. [2]

Recommendation Systems use various classification rules which aim to classify the patterns of the user(s) into various classes to optimize the search and results of the system. The most upcoming and rapidly growing branch of Recommendation Systems is "Mobile Recommendation Systems". Almost half of the population of the world owns a smartphone and thus getting personalized feed on their phones is an interesting and new attraction which has led to the rise of Mobile Recommendation Systems.

A. *Content-based Filtering*

They are more subjective and description based i.e. every item to be classified has a different description and suggestions depend upon on the user's profile only. There are keywords/classes that describe and entity to be studied which indicates what type of entity it is and then the type is added to the existing user profile and then suggestions compromise of entities from the identified type and type similar to it.

After a user profile is made various weighted factors are calculated, where weight depicts the importance, using various techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks which determine the possibility of a user showing interest in any particular entity.

A problem with Content-based filtering is that it is only "Content" based, i.e. only one type of content can be classified. For example, if a system is filtering news related to/according to a user then the system should also recommend movies or music or articles pertaining to the person as well.

Also, content-based filtering is solely restricted to the user and the history thus it does not take into account future predictions (i.e., if the preferences of the user change) and the surroundings and peer of the user. [4]

.

B. *Collaborative Filtering*

Collaborative Filtering, on the other hand defines a group of users into a type on the basis of matching criteria and similarity to other users. One advantage is that because it typecasts users and not objects thus it can recommend a wide range on entities to a group of users after identifying the type of users.

It assumes that a group of users who have shown interest in an entity in the past will also show interest in the future and keep doing so until the whole type of user is changed. Thus, a recommendation for a single user can be seen as something recommended by his/her group to him/her.

Most social networks use Collaborative Filtering because of the incorporation of "Gamification". The social networks want the users to be stuck to their websites/applications and hence they always show results which are related to their social group and connections.

C. *Hybrid Recommendation System*

Hybrid, as the name suggests, is the amalgamation of various types of recommendation systems into one. It can provide a more accurate range of recommendations as it depends on multiple systems and can compute various factors in one go.

It can be done in a number of ways: by designing different content-based and collaborative filtering systems and then combining the results and present as one; by implementing content-based classification into a collaborative one or vice versa.

Seven hybridization techniques:

- Weighted: Combination of weights from different recommendation components.
- Switching: Intelligent switching of recommendation systems depending upon the type of request.
- Mixed: Recommendations from different recommenders are presented together.
- Feature Combination: Different pointers from various recommendation engines are selected and parsed into one system.
- Feature Augmentation: The output of one recommendation system using "Feature Combination" is used as the input for another system which then computes the remaining features.
- Cascade: Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
- Meta-level: The output of one recommendation system is used as the input for another.

D. *Vector Space Model*

Because Vitae follows a textual approach in mapping the skills of a specific user with the jobs of his/her respective industry, Vector Space Model was the best choice to accomplish the mapping with minimal scope for error.

Vector Space Model, also called as the Tf-Idf representation, the vector space diagram is a text mining algorithm which represents documents as vectors and defines the similarity index in terms of the angle that each vector makes with the respective axis. [7]

It determines the importance of keywords in a document mapped with other documents of the same kind. Because it takes a collection of documents each time it determines the type of the document, it becomes more and more intelligent with time as more documents are added to a collection of specific type of documents. [6]

The calculations for Tf-Idf for Skills vs Jobs are as follows:
Skill = s,
Job = j,
Set of Jobs = J

Tf (s, j) = number of times the Industry (i) of skill ‘s’ matches with the Industry of the skills present in job ‘j’.

$$TF = \frac{sF}{n}$$

Here sF is the frequency of the Industry (i) of skill ‘s’ and ‘n’ is the total number of skills.
IDF (s, J) measures the importance of the Industry (i) in the set of Jobs ‘J’.

$$\text{IDF}\ (s, \text{J}) = 1 + \log_e \frac{\text{N}}{\text{JF}}$$

Here, N is the number of jobs in the set and JF is the number of jobs in which the skills ‘s’ appears.
We add 1 for normalization, in case log is 0.

The final weight for each job is calculated by

$$\text{TF-IDF (s, j, J) = TF (s, j) x IDF (s, J)}$$

E. The decision of NoSQL database (MongoDB) over SQL databases

For a dataset of so many entries with multiple preferences to take care of, a fast, flexible, and cleaner database was required.

The first choice was SQL, but running multiple cascading SQL queries was consuming a lot of time even if it was in milliseconds.

Yishan Li and Sathiamoorthy Manoharan, in their work, have outlined various NoSQL databases’ performances and compared them with SQL databases. NoSQL databases took almost half the time taken by SQL databases to read values. [8]

Since MongoDB stores data in a horizontal format in the form of documents, and supports embedding, it was chosen to be the primary database for this application. MongoDB is a document-oriented database written in C++ [9].

F. How is Vitae different?

Vitae follows a textual approach in comparing the user’s skills with the jobs. As soon as a user registers with the app, adding his/her skills to the system, the user is assigned an Industry (which is calculated on the basis of the skills that the user has provided). This industry is derived upon the basis of the probability of the user belonging to a particular industry which has the respective skills.

When the user sends call for the “Recommended Jobs”, the system maps the user’s skills with the jobs present in the Industry which was assigned to the user. This approach makes the system faster and more reliable as the industry is solely based upon the user’s preference which then affects the results of the “recommended” job. Thus, staying true to its name Vitae is exclusively based upon the user’s skills and abilities with a few other factors contributing to the final weighted factor.

III. Problem Definition

A. Problem Formulation

The Recommendation system solves 2 problems: 1) Display the list of suitable jobs for a specific user with distinct skill or check if a job is suitable for the user and 2) To display the most suitable candidate, for a job, to the recruiter.

Both of these are solved in a similar manner, Tf-Idf representation. For the former problem, the user is the entity to be mapped whereas for the latter, the job is mapped against the candidates.

The Recommendation system calculates the Tf-Idf for each skill of the user against each skill of every job found. Several other factors are taken into account to calculate the weighted factor with Tf-Idf occupying the highest weight.

B. Terms Used

The mapping of the User object, to be mapped with the jobs, is done based on the following features:

Education: The highest qualification level of the user.
Industry: The field of the user which is determined by his skills.
Skills: The skills of the user as entered.
Location: The current location of user.
Position: The current position of the user.
Tf-Idf: For each skill of the user.

The mapping of the Job object, to be mapped with the user, is done based on the following features:

Industry: Industry that the job belongs to.
Skills: Skills of the job.
Experience: The experience required for each job.
Location: The location of the job.
Position: The position offered.
Salary: The salary offered for the job.
Factor: The weighted factor of each job.

C. Assumptions

1. It is assumed that the user will acquire the skills, that are not present in the recommended job, in the coming future.
2. It is assumed that the user will be interested in the most suitable job that belongs to his/her industry only.

IV. Methodology

A. Feature Selection

The Recommendation System needs various features to be able to recommend a job or a candidate.

The features for the User are: Education (Highest Degree), Industry, Skills, Location and Current Position.

The features for the Jobs are Industry, Skills, Experience, Location, Position, Salary.

B. *Working of the RESTful Web Server*

The Web Server is based upon the Jersey (JAX-RS) framework which enables it to receive, handle and send RESTful API calls.

The Web Server uses MongoDB as the Database instance and the Morphia driver for MongoDB is used to persist all the entities and to carry out all the necessary queries.

The Web Server can receive calls from any REST interface, be it a REST client or Android/iOS device. The call has a specific path which points to a distinct method in the respective class.

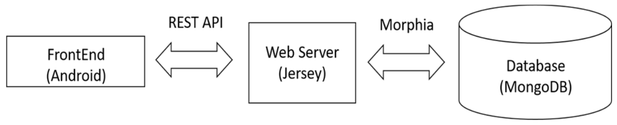

Fig. 1. Communication of the Web Server with other Modules

C. *Working of the Recommendation System*

Upon receiving the API call to fetch the recommended jobs for the user with "ObjectId" passed in the parameters in the form of:

API Call: 34.196.71.31/job/rest/jobs/recommended?user_id= 590899f28c39c338765cb83b

Here, (34.196.71.31) is the elastic IP of the Amazon EC2 Instance where the Web Server is hosted.

The "job/rest/jobs/recommended" refers to the path of the method to calculate the recommended jobs.

The "user_id=590899f28c39c338765cb83b" is the Form Parameter where the Id refers to the unique Id of the user as stored in the database.

Upon receiving the call Morphia driver fetches Categorized Jobs for the specific field from the MongoDB database. Each categorized job has a set of skills.

The user object (containing user skills) with the categorized job list (also with skills) is added to the recommendation system. Tf-Idf is calculated for each skill of the user against each skill of each categorized job, creating an impression of comparing a word with a set of documents having that word. The jobs are returned sorted in decreasing order of Tf-Idf. After that various other classifiers are applied to the data which contribute to the final weighted factor.

The jobs are then again sorted in decreasing order of final factor and returned to the Frontend.

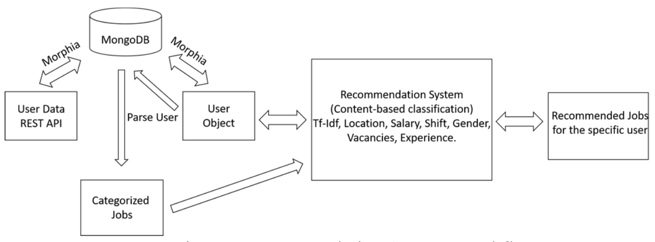
MongoDB
Morphia
Morphia
User Data REST API
Parse User
User Object
Recommendation System (Content-based classification) Tf-Idf, Location, Salary, Shift, Gender, Vacancies, Experience.
Recommended Jobs for the specific user
Categorized Jobs

Fig. 2. Recommendation System Workflow

Example: There are two Jobs: 1) Job A; 2) Job B and two Users: 1) User A; 2) User B

Job A has the skills – U, X, Y, Z
Job B has the skills – V, W, X, Z

User A has the skills – U, Y, Z
User B has the skills – U, W, Z

Factor (User A, Job A) = $[\Sigma \text{ Tf Idf } (\text{User A}, \text{JobA}, \text{Job A\&B})]$
Factor (User A, Job B) = $[\Sigma \text{ Tf Idf } (\text{User B}, \text{Job B}, \text{Job A\&B})]$

Factor (User A, Job A) = 0.3715
Factor (User A, Job B) = 0.25

Hence, the User A is closer to the Job A. Similarly, the User B is close to the Job B.

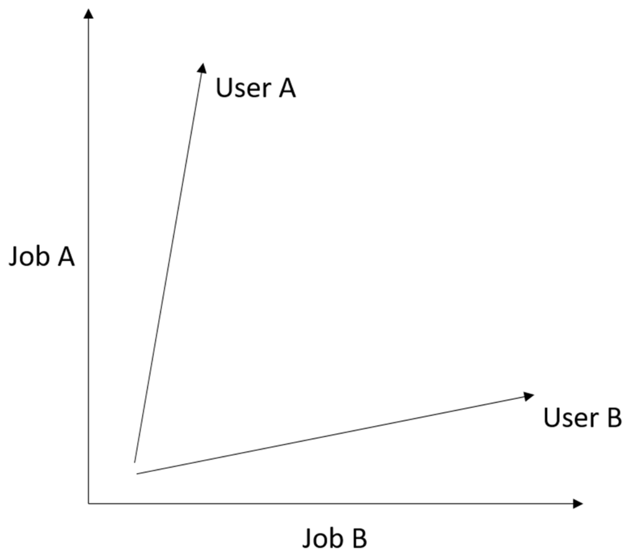

Fig. 3. Vector Space Model for Users (A&B) and Jobs (A&B)

D. *Working of the Android Application*

Android Studio is used to design the frontend of the application. The minimum Android SDK level is set as 14 (Android 4.0 Ice Cream Sandwich). All external libraries: Retrofit; Google Play Services (for all Google Developer features); Google Firebase etc. are compiled in the ".gradle" file.

Layout for each view, Activities and Fragments and Dialog Boxes, is designed and then imported in the main class. As the app is first started the MainActivity.java handles the flow of control via Intent(s) and Fragment Manager.

It incorporates various APIs and technology stacks. It would also serve as the only interaction the user would need to have to get all work done.

At the first launch the Loader Activity is added to View and contains two fragments, Splash and Login. After the Login, the user can either submit their resume and profile or create one.

After that control passes to the “Main Activity” which has three fragments with focus on the “Search Fragment”. The user can switch to the rest of the fragments using Tab View Pager which is a Sliding Tab Strip.

The Main Activity is the one that calls the Retrofit Model for REST APIs. It contains of a Singleton (Retrofit Instance), a Retrofit Caller (which has all the API calls to be used) and POJOs (The Pojo classes of all the Objects). Main Activity and Retrofit Model communicate to send and receive API calls.

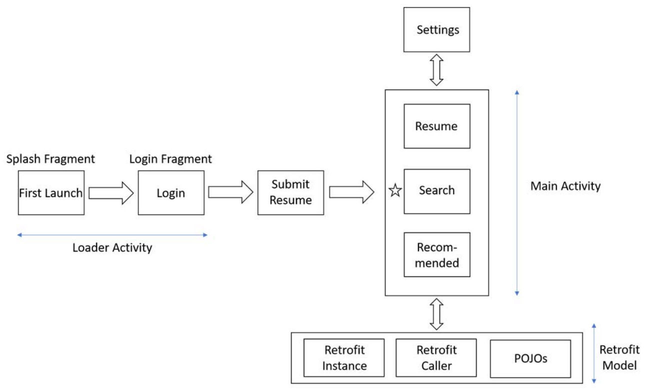

Fig. 4. Android Application Workflow

V. Results And Discussions

The dataset was created by adding 30 Industries, 147 Categories, 2232 Skills and 50 Jobs incorporating various Skills.

All services, MongoDB and Tomcat7 were initialized on the Amazon EC2 Instance.

REST API Client (Postman) was setup to test the calls and the Recommendation System. The first API call was containing user details is sent to the Web Server which passes the information to the Morphia driver which persists the “T_Users” Collection (Table for SQL) with the User Document (Row for SQL).

The API call for recommended jobs is sent to the server with the User Id as parameter. The system first fetches the list of jobs which have the same industry code as the user. This list of jobs is then sent to the Recommendation System with the user document where the userSkills are mapped with the jobSkills. After calculating tf, idf and tf-idf for each skill, the weighted factor is calculated giving equal probability to each jobSkill. The job with the highest weighted factor is the most “suitable” job.

The modules were tested again with more calls and data and successful results were recorded. The whole workflow is impeccable and works seamlessly. The average call response time recorded with MongoDB was 150ms whereas average call response time for SQL and Hibernate Framework was 900ms. All the calls were made through Postman REST Client and now have to be shifted to the Android interface.

After testing against a plethora of entries, the RESTful API Framework was designed on the Android Application. LinkedIn login was incorporated to maintain the veracity of the details of the user. LinkedIn login requires the developer to register with them, granting them an Application Id and Security Key. Users can Login via Email as well but LinkedIn Login offers various features such as time efficiency, verification of details via the OAuth 2.0 Library and linking of LinkedIn data with the user profile.

The Retrofit Library enables the application to make calls from the main Thread itself which is not allowed by normal HTTP calls. Frontend can make API calls and can receive them without any occurrence of Exception or the Application undergoing a forced stop. The frontend is still in the initial stage and there is a lot of scope for improvement in terms of CPU cycles and multi-threading.

The Web Server is hosted on the Singapore Server by Amazon Web Services and will continue to run for another year free of cost. The Recommendation system results were mapped and found satisfactory. The Android Application is in the MVP (Minimum Viable Product) stage and can be released to the App Stores for user discretion.

VI. Conclusion

The proposed framework for Job Recommendation System is aimed at bridging the gap between employees and employers. The responses returned by the system were acceptable within a certain margin of error. Further studies into the topic can help enhance the precision of the system.

Future prospects include

- Implementing Partner Program of LinkedIn, which requires partnering with LinkedIn for future projects. Using this service developers can get access to every API and data present for any specific user. This will help make skill and endorsements mapping easier and verified.
- Adding web crawling to determine the location with most demand for a certain Job Description or a certain user with distinct skills.
- Classifying every job as well as a resume into classes before recommendation using Named Entity Recognition to make results more accurate.

The application has the potential to stand financially in the market but needs to be improved a bit and then presented to Investors looking to invest in such projects. The Android application is ready for release to the Google Play Store. But for the idea to be financially viable, financial aid is required to kickstart it.

References

[1] R. Munger, “Technical communicators beware: The next generation of high-tech recruiting methods.”, IEEE Trans. Professional Communication, vol. 45, pp. 276-290, 2002. (*references*)

[2] Anika Gupta, Dr. Deepak Garg "Applying Data Mining Techniques in Job Recommender System for Considering Candidate Job Preferences", International al Conference on Advances in Computing, Communications and Informatics (ICACCI) 2014.

[3] Deepali V Musale, Mamta K Nagpure, Kaumudini S Patil, Rukhsar F Sayyed "Job Recommendation System Using Profile Matching and Web-Crawling.”, INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH AND ENGINEERING TRENDS.

[4] Michael J. Pazzani and Daniel Billsus "Content-Based Recommendation Systems", The Adaptive Web, LNCS 4321, pp. 325– 341, 2007.

[5] Al-Otaibi, Shaha T; Ykhlef, Mourad. "Job Recommendation Systems for Enhancing E-recruitment Process", Proceedings of the International Conference on Information and Knowledge Engineering (IKE); Athens: 1-7. Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). (2012).

[6] R *aghavan, Vijay V; Wong, "A Critical Analysis of Vector Space Model for Information Retrieval", S K M. Journal of the American Society for Information Science; New York, N.Y. 37.5 (Sep 1, 1986): 279.*

[7] Amit Bagga, Breck Baldwin, "Entity-Based Cross-Document Coreferencing Using the Vector Space Model", COLING '98 Proceedings of the 17th international conference on Computational linguistics, Montreal, Quebec, Canada — August 10 - 14, 1998.

[8] Y. Li and S. Manoharan, "A performance comparison of sql and nosql databases," in Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on, Aug 2013, pp. 15-19.

[9] K. Chodorow and M. Dirolf, MongoDB: The Definitive Guide. O'Reilly Media, September 2010.