Building Recommendation Engine

Hemang Jethava - 1644002
Algorithms and Optimisation for Big Data
School of Engineering and Applied Science
Ahmedabad University

***Abstract*—Basic idea of the report is to design such a Recommendation engine in which following two modules must be covered. A module that reads users profile and suggest a career path in terms of skill set to be acquired. A module in which user enters a career goal and based on this career goal and other related information the platform suggest a career path. Algorithms and results will be shown.**

I. Introduction

What is Recommendation Engine: Recommendation engines are nothing but an automated form of a shop counter guy. You ask him for a product. Not only he shows that product, but also the related ones which you could buy. So, does our recommendation engines.[3] They are well trained in cross selling and up selling. This System is used in linkedin,facebook and there are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners and Twitter pages[4].

Personalization system such as recommender system attracted the interest of many researcher and practitioners. Many techniques for suggestion of career path and job recommendation have been developed and applied. These includes the one of the possible approach for career path recommendation system.

a) Algorithms: Now lets come to the special class of algorithms which are tailor-made for solving the recommendation problem. There are typically two types of algorithms Content Based and Collaborative Filtering.

A. Collaborative-Based Algorithm

Collaborative filtering methods are based on collecting and analyzing a large amount of information on users behaviors, activities or preferences and predicting what users will like based on their similarity to other users[5].

A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an ”understanding” of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems.

there are several types of collaborative filtering algorithms[1]:

User-User Collaborative filtering: Here we find look alike customers (based on similarity) and offer products which first customers look alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every customer pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallel system.

Item-Item Collaborative filtering: It is quite similar to previous algorithm, but instead of finding customer look alike, we try finding item look alike. Once we have item look alike matrix, we can easily recommend alike items to customer who have purchased any item from the store.

This algorithm is far less resource consuming than user-user collaborative filtering. Hence, for a new customer the algorithm takes far lesser time than user-user collaborate as we dont need all similarity scores between customers. And with fixed number of products, product-product look alike matrix is fixed over time.

B. Content-Based Algorithm

If you like an item then you will also like a similar item Based on similarity of the items being recommended, It generally works well when its easy to determine the context/properties of each item. For instance when we are recommending the same kind of item like a movie recommendation or song recommendation.

II. Preprocessing of Data

First, Data cleaning would be the essential part to perform. We Clean the JSON files by removing the non-ASCII characters and Converts the cleaned JSON files to CSV Format. Now, we take user input for goal and suggests path and Selects profession and User then suggests path.

Cleaning of data must be Done before processing of the Data. Data cleansing, data cleaning, or data scrubbing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

III. My approach

There are plenty of ideas to come up with the solution for recommender system. Thus the basic idea for this report is as follows.

In our recommendation algorithm, we will maintain a number of sets. Each user will have two sets: a set of skills the user have, and a set of skills the user wants . Each skill will also

have two sets associated with it: a set of users who has that skill, and a set of users who does not have it despite being in the same field. During the stages where recommendations are generated, a number of sets will be produced - mostly unions or intersections of the other sets. We will also have ordered lists of suggestions and similar users for each user.

To calculate the similarity index, we will use a variation of the Jaccard index formula[6]. the formula compares two sets and produces a simple decimal statistic between 0 and 1.0:

$$J(A,B) = \frac{A \cap B}{A \cup B} \tag{1}$$

The formula involves the division of the number of common elements in either set by the number of all the elements (counted only once) in both sets. The Jaccard index of two identical sets will always be 1, while the Jaccard index of two sets with no common elements will always yield 0. Now that we know how to compare two sets, let us think of a strategy we can use to compare two users.

As discussed earlier, the users, from the systems point of view, are three things: an identifier, a set of skills, and a set of desired skills. If we were to define our users similarity index based only on the set of their skills, we could directly use the Jaccard index formula[3]:

$$S(U1,U2) = \frac{L1 \cap L2}{L1 \cup L2} \tag{2}$$

Here, U1 and U2 are the two users we are comparing, and L1 and L2 are the sets of skills that U1 and U2 have, respectively. Now, if you think about it, two users have the same skills are similar, then two users' desired skills should also be similar. Now, for given problem jaccardian indices approach is different than the above equation,

First of all what career goal is right for the user or what the user wants is being analysed. It then makes a list of all the skills that users have in that profession. Then we form a matrix S of each user and their skills for that Career. Formation of Matrix T of pairs of skills and how many time they occur together will be done. After this the Jaccard Index of all pairs of skills is calculated and stored in a matrix J where J(i; j) is the Jaccard Index of Skills i and j. The Jaccard Index of two values are calculated as follows:

$$J = \frac{AB}{A + B - AB} \tag{3}$$

where, A: Number of times that A occurs B: Number of times that B occurs AB: Number of times that AB occur together. Then the Skills having the top 3 Jaccard Indices are suggested to the user Thus this idea is almost similar to previously discussed idea for module 2 of first idea but this approach can be applied on both the Modules.

IV. Implementations and Results

The collaborative Approach Using Jaccard indices approach will give the following outcomes for our recommender System.

A. For Module 1

Figure 1 is a Result of implementation of the Jaccard indices method for suggesting skills to the User.

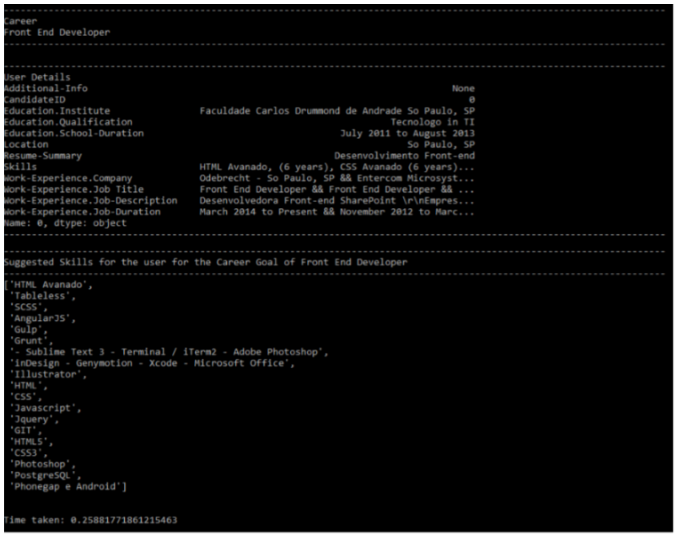

Fig. 1. Fig 1: Showing Suggested skills for the User

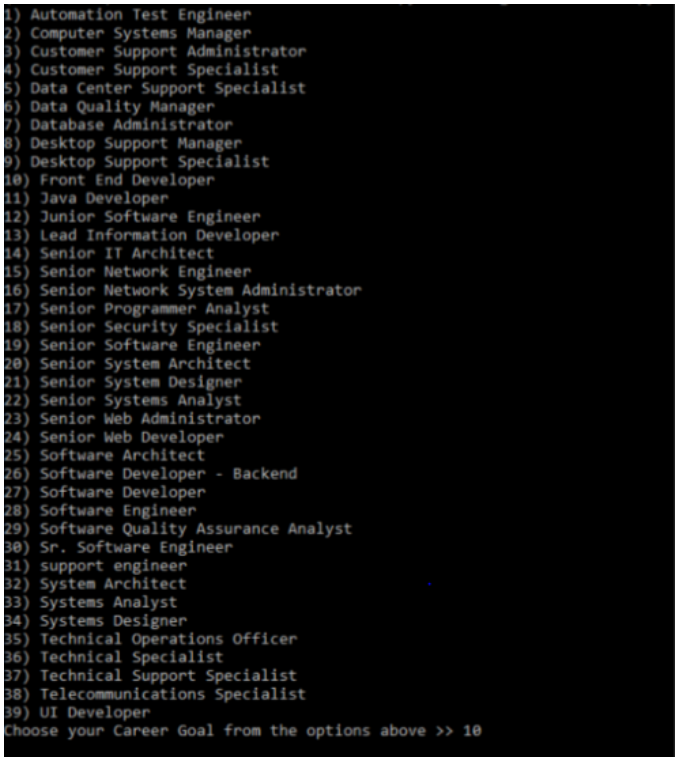

Fig. 2. Fig 2: Giving options to the users for career path

B. For Module 2

Figure 2 is given options to the users to select for a career path in their respective fields.

Figure 3 is a Result of implementation of the Jaccard indices method for suggesting career goals to the user after choosing the career.

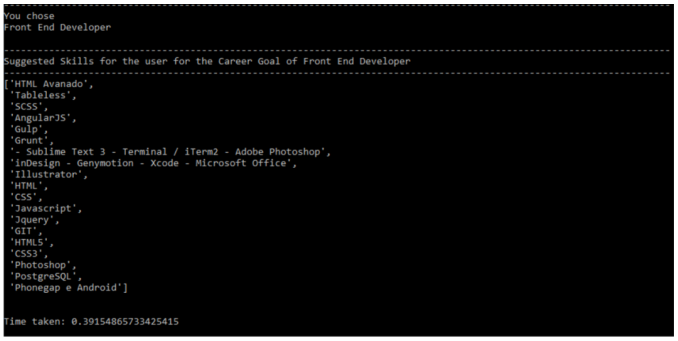

Fig. 3. Fig 3: Showing Suggested Career paths for User

The Jaccard indices Method has a time complexity

$$O(mn^2)$$

Where m is the number of users and n is the number of skills is having a very high efficiency with good accuracy with a better and more connected career path to the user.

Github Repository for the codes and this report is available at [2]

V. Conclusion

Memory-based collaborative recommendation engine algorithms can be a pretty powerful thing. The one we experimented with in python may be primitive, but its also simple to understand, and simple to build. It may be far from perfect, but robust implementations of recommendation engines, such as Recommendable, are built on similar fundamental ideas.

References

[1] AnalyticsVidhya. Recommendation problems.
[2] H. Jethava. GitHub.
[3] TopTal. Recommendation engine.
[4] Wikipedia. Recommender system.
[5] Wikipedia. Collaborative filtering.
[6] Wikipedia. Jaccard Index.