Career Path Suggestion System

Rupande Shastri
AOBD Final Exam
BTech, ICT (1401102)
SEAS, Ahmedabad University

***Abstract*—Many automated recommendation systems are available for suggesting career options to people based on their skills, education and other relevant information but due to the rigidity of type of information these systems take and process, they are not popular among a majority of users. Hence arises a need for a career path suggestion system that takes as input the whole profile or resume of a user and gives career path suggestion based on the input given to it. A model of one such career path suggestion system is described in this report. This system takes as input a user profile similar to a LinkedIn profile and after extracting all relevant information and processing, predicts a number of job titles that the user may aim for based on their skills and also suggests the skills (if any) they need to acquire in order to attain the job titles suggested. The system is first trained by a set of user profiles before it can make predictions.**

Keywords*—*Career Path Suggestion, Recommendation Systems, KNN Classification, Weak Classifiers, Data Cleaning.

I. Introduction

Until very recently, the process of seeking career advice has been tedious and time consuming, and yet inefficient. Even with automation in many fields, an automated solution to the problem of career path suggestion is not widely accepted. This is not due to a lack of systems online - there are numerous online and offline systems that dole out career options to their users. While some of these systems ask information as little as the user's skills (proven by some qualification certificate) in relevant fields, some go as far as to make the user give aptitude tests ranging across a variety of fields. In most of these systems, the user has to take the time to give their information in the format required by the system. This need for a fixed format makes them unpopular among seekers of career advice, some of whom do not have the time for it and others who are looking for suggestions based on a more personalized set of factors. A general (extendable) list of these factors include skills in various fields, education, other qualifications, achievements, location and their own areas of interest. This information is provided in their LinkedIn profiles and resumes. A career suggestion system that uses these profiles to suggest job options can prove to be more popular as it is flexible in terms of input and thus significantly lessens the effort needed from the user's side.

Another aspect of a career that is often ignored while job-seeking is improvement of the skill-set. Most people settle for jobs that are available based on their current skill set and make progress in their career solely by gaining work experience. This approach works as they gain skills along with work experience but it can be made better if the person knows the opportunities open to them by acquiring a few other skills. This career path suggestion system is modeled to include job opportunities that a user can attain by gaining a few more skills, which are also listed with the job suggestions. The challenges here are:

- To extract relevant information from the semi-structured user profiles
- To model a system that gives near-accurate predictions when the size of the information varies for each user and when data for some relevant fields is not provided

II. System Overview

Treating the system as a black box, we have the input as a user profile and the output as a set of job opportunities with a corresponding list of skills to be acquired (if any). This can be achieved by first processing the user profile to extract relevant information, then processing the information to get suggestions. It can thus be divided into two modules: Data Extraction and Job Suggestion.

The task of the first module is to parse the user profile, ignoring irrelevant words and extracting relevant information. The extracted information needs to be structured in order to make it available for processing. The second module takes the structured information as input and processes it to generate suggestions. The processing involves comparison of the input information with corresponding relevant information of other users.

A. Data Extraction

Using profiles and resumes leads to difficulties in structuring the information given. However, parts of it can be fixed by simple text processing. The data can be segregated into various factors common to each resume - candidateID, skills, location, education, institute, company, job titles (from work exprience). Other factors that are harder to extract are interests,career goals, projects and work experience. The data that can be segregated or Class-I data can structured based on their respective factors and used directly for processing. The other data or Class-II data needs text analysis techniques before it can be used. The current implementation of the system uses only Class-I data separated by their corresponding factors.

B. Job Suggestion

The method to suggest jobs is based on a human tendency of seeking advice from peers and seniors in the same field of work, education and with the same skills. Structured user data is fed to a set of classifiers. These classifiers are trained

by a significantly large number of previous user data. Job titles are suggested from the pool of job titles of the trained user data. Skills to be acquired are taken from the skill-sets of the users who have the suggested job titles.

C. KNN Classifiers

K-Nearest Neighbours(KNN) classifiers are used in our current implementation. For a N featured dataset, the KNN model keeps information about the Euclidean distance d_{ij} for each $i \neq j$ in the training data along with their labels. At the time of classification, it checks for k data points with minimum distance from the given data point in the N-dimensional space. The label assigned to the data point will be the majority of labels of these k data points. In our case, the input to these classifiers is user data from a profile. The k-nearest neighbours found will be the data of users with profiles similar to the given profile. This is consistent with what we intend to do, which is to find job titles which suit the features in the given user profile.

III. System Model and Working

The modules stated in the previous section are to be implemented separately on the given dataset.

A. Data Extraction

User profiles are fit to a table - useful columns containing separated Class-I data and other columns containing Class-II data. The columns that are to be used get converted into lists and relationship matrices. A list contains each unique element of the column.

F1 (candidateID)	F2 (skill)	F3 (company)	O1 (career goals)	O2 (projects)
0	Skill_1, skill_2.. skill_n	Com_1, com_2,...com_m	Long sentences	Long sentences
1 . . .	Skill_2,...skill_m . . .	Com_5,...com_m . . .	More Long Sentences	More Long Sentences
n	Skill_3, skill_5,.. skill_m	Com_2, com_7,.. com_n	Long Sentences	Long Sentences

F2 (skill)
Skill_1
Skill_2 . .
Skill_n

User	Skill_1	Skill_2	Skill_n
1	1	1	1
2	0 . . .	1 . . .	0 . . .

A relationship matrix of factor F is defined between factor list L_F and User Data Table as:
if element f_i in L_F belongs to j^{th} factor set in User Data Table
then $x_{ij} = 1$
else $x_{ij} = 0$
This means that the relationship matrix contains factor set for each user. The above activity is done for every factor in the User Data Table. Supposing the number of factors are n, we have n lists and n relationship matrices that represent user data.

B. Job suggestion

Taking n lists and n relationship matrices as input, one of the factors being job titles, we augment all $n-1$ relationship matrices except the *job titles* matrix.

User	Skill_1	Skill_2	Skill_n	Com_1	Com_2...	Com_n...
1	1	1	1	1	1	0
2	0 . . .	1 . . .	0 . . .	0 . . .	0 . . .	0 . . .

This becomes the feature matrix X to train the classifiers. If the sum of all elements of all lists (except *job titles*) is N and we train using m users' data, the dimensions of the feature matrix will be mxN.
The labels are given by Y where Y is the relationship matrix of *job titles*. We use K-Nearest Neighbours(KNN) classifiers to serve our purpose. Let the number of elements in the *job titles* list be J. Then J binary classifiers will be used to label whether a given feature vector is suited for the job.
Each binary classifier will have as training input a matrix of size mxN and a label y to determine whether the features suit the job. It is assumed here that if a user having a feature vector X has a job y then the features in X suit y.
These classifiers do not provide the information we require in themselves, therefore we build an ensemble classifier of the J weak classifiers. When fed with an input of a feature vector for a user of size 1xN, the output of the ensemble classifier would be a 1xJ vector with binary elements (All the matrices and vectors used here have binary elements).
The indices with 'ones' in the output vector are stored in a set Pj. A set Hj containing indices of jobs the user already has in their *job titles* factor set is subtracted from Pj to give a set Rj. The indices in Rj are mapped to the *job titles* list to give a set of recommended jobs for the given user.
The users having the jobs in Rj are then found in the relationship matrix of *job titles*. They are called reference persons for the job. For each job, the indices of the combined skill set of its reference persons Ps is subtracted by the indices of the skill set of the user in question Hs to give Rs. This is mapped to list of skills to give recommended skills. Hence, each job will have its Rs. This set can be null if the user has all the skills required for the job.

IV. Results

After extracting 3 relevant factors from the dataset given, the algorithm was tested on a training data of 400 samples with 4470 features in the feature matrix and 2000 job titles. The MATLAB results are shown below:

```
>> jobs(recommended_jobs)

ans =

    'Technical Support Engineer/ System Administrator'
    'DBA'
    'SQL Server Database Administrator'
```

```
>> skills(recommended_skills(1,:))

ans =

    'CSS3 (5 years)'
    'oracle applications'
    'Filemanager'
```

Recommended skills for the first recommended job are shown below the list of recommended jobs.

The time taken while varying number of job titles, number of features and number of samples is given below:

Number of Js	Time taken (s)	Number of Ns	Time taken (s)	Number of ms	Time taken (s)
2000	66	4000	18	400	11.5
1000	36	2000	14	200	10.8
500	19	1000	11.5	100	9.6

From the above data we can see that the most significant change in time taken is due to change in number of job titles, and the change is almost linearly proportional. Since there are three variables, change due to which has the upper bound of a linear function, we can say that the time-complexity of this algorithm has an upper bound of $O(n^3)$.

V. Other Approaches and Future Work

Other approaches to this problem can be:

- Latent Semantic Analysis - Relations between documents are defined by a term-frequency matrix. Compute the Singular Value Decomposition of the matrix and use it to form clusters or match keywords based on semantic similarity.
- Fuzzy Logic can be used to compute ratio of similarity between two arguments.
- Career paths can be modeled using a tree. One can traverse it to predict a complete career path.

Limitations of the current version of the algorithm:

- The number of jobs predicted cannot be defined.
- There is no definite way to rank jobs based on similarity to profile.
- The matrices generated and used by the algorithm contains lots of redundant data.
- Class-II information of a profile is relevant but cannot be used, hence we are dealing with significant information loss by our data extraction method.

Future work will be on these limitations using dimensionality reduction for redundant data and Latent Semantic Analysis for Class-II information. To rank jobs based on similarity, fuzzy logic can be applied.