LinkedIn Career Recommendation System

Nisarg Tike
Roll No: 1401070

***Abstract*—LinkedIn, the world's largest professional network uses horizontal collaborative filtering known as *browsemaps* to showcase relationships between pairs of items according to the wisdom of crowd. This is used for job suggestions and other related aspects. In this report, an algorithm is proposed for career path recommendation in terms of skillset. This paper uses the concept of transformation matrix and clustering. Given a training data of users having a set of skills and corresponding job position, the algorithm can suggest a career path for some arbitrary set of skills.**

Keywords:Transformation matrix, Skill space, Job space, Clustering

I. Introduction

In this paper, a career recommendation system is presented which is designed for professional platforms like LinkedIn. The system reads user profile and based on the parameters learned from training data, it is able to recommend a skill set that matches the user's career goal. The proposed algorithm is an online version that continuously learns and adapts to new data. The rest of the paper is organized as follows. Section II states the assumptions. Section III describes the detailed approach to the problem. Section IV discusses the algorithm and its efficiency in terms of complexity. In Section V, results are showcased. Finally, Section VI concludes the paper.

II. Assumptions

Before attempting the problem, certain assumptions were considered appreciating the challenge and depth of the problem. Following are the assumptions.

i) The training data and the test data is cleaned before feeding it to the system.

ii) Every user in the training set has exactly one job position. However, he/she can have variable number of skills. There is no restriction that two users with same set of skills have a common job position.

III. Approach

The problem at hand does not contain any numeric representation. Therefore, the foremost task is to represent a text data into numeric form. If there are N_s skills, each skill can be mapped to a positive integer between 1 to N_s. Likewise, if there are N_j jobs, each job can be mapped to a positive integer between 1 to N_j. But it does not give us a mathematical significance by merely mapping text to integers.

Every skill can be represented as a vector in 3-dimensional space. We call the collection of all such skill vectors as *Skill space*. Similarly, every job can be represented as a vector in 3-dimensional space. We call the collection of all such job vectors as *Job space*. It is intuitive to understand that two similar skills will be located near each other in the respective space. For instance, if 'C language' is located at $[1\ 4\ 6]^T$, 'C++ language' will be located at $[1\ 3\ 5]^T$. But a skill like 'Project Management' which has less correlation will be located at $[40\ 23\ 10]^T$. In general, a set of related skills will be close to each other and a set of related jobs will be closer to each other in the respective space.

Next, we define a transformation matrix A(3×3) that maps a set of vectors in 'skill space' to possible job vectors in 'job space'. We define a 'skill matrix' $S_i(3\times\ k_i)$ for a user i having k_i skills. Each vector in S_i is a skill vector. Also, a 'job matrix' is defined as $J_i(3\times\ k_i)$ for user i. Each vector in J_i is an approximated vector in 'job space' that is close to the actual job position of the user.

$$AS_i = J_i$$

We randomly initialize A such that it is invertible. Also we randomly initialize all the skill vectors. Job vectors are not initialized. It is important to note that the values of the vectors are insignificant and only the relative distances matter. It is highly unlikely that randomly initializing vectors will make sure that related skills are nearer. But after training, the skill vectors and job vectors will realign to the desired locations. We infer that if a user has ki skills, they are related and so the skill vectors must contract towards a weighted mean. After updating, J_i is computed using. Note that J_i is the approximated job vector. The actual job vector is realigned using weighted mean. This makes sure that related jobs are clustered together.

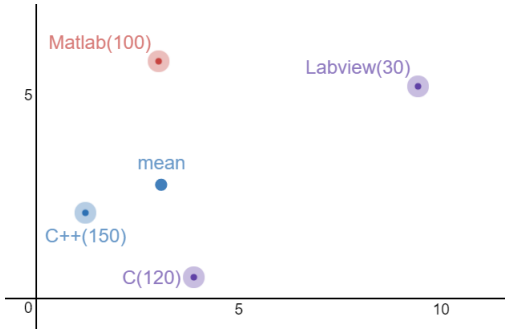

Fig. 1: Skill Space

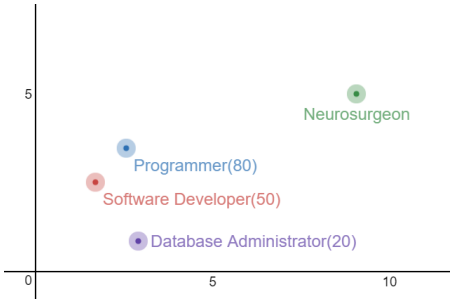

Fig. 2: Job Space

IV. Algorithm

A. Training Pseudo Code

Following is the pseudo code of online version for training and aligning skill vector and job vectors

Algorithm 1 Align skill vectors and job vectors

```
Randomly initialize S
Initialize Scount to 0
Initialize J to 0
Initialize Jcount to 0
Randomly initialize A such that it is invertible
for every user i in training set do
    read profile [Ui, UJi]
    increment Scount
    update all skill vectors
    build Si using Ui
    Ji = A*Si
    if job is new to the program then
        J(:,Uji) = mean(Ji);
    else
        find vector in Ji nearest to J(:,Uji)
        update the job vector according to weighted mean
    increment Jcount
```

B. Module 1

Following is the pseudo code for module 1

Algorithm 2 Recommend skills based on partial knowledge

```
Read Ui
for every skill vector Si in Ui do
    display two nearest skills in skill space
```

C. Module 2

Following is the pseudo code for module 2

Algorithm 3 Recommend skills based on career goal

```
Read Uji
xHat = inv(A)*J(:,Uji)
find nearest skill x to xHat
threshold = 3*norm(x-xHat)
for all skills i in skill space do
    if norm(S(:,i)-x) is less than threshold then
        display S(:,i)
```

D. Efficieny

The proposed algorithm is online and continuously updates the skill vectors and job vectors with new training data. The time complexity for algorithm 1 that trains data is O(n_j) for each candidate where n_j is the total number of jobs in job space. The dominant cost of algorithm 1 lies in finding the nearest job vector from the approximated job vector.

Algorithm 2 reads candidate profile and recommends the related skills for the candidate to pursue. The time complexity of this algorithm is O($k_i n_s$) where k_i is the number of candidate's current skill and n_s ($n_s >> k_i$) is the total number of skills in the skill space. The dominant cost of this algorithm lies in finding the nearest skills in skill space.

Algorithm 3 suggests skills according to the candidate's goal. The time complexity of this algorithm is O(n_s) where n_k is the total number of skills in the skill space. The dominant cost of this algorithm lies in finding the nearest skills in skill space.

V. Results

The above algorithm was implemented in matlab and was tested on cleaned candidate profile dataset. Following are the results.

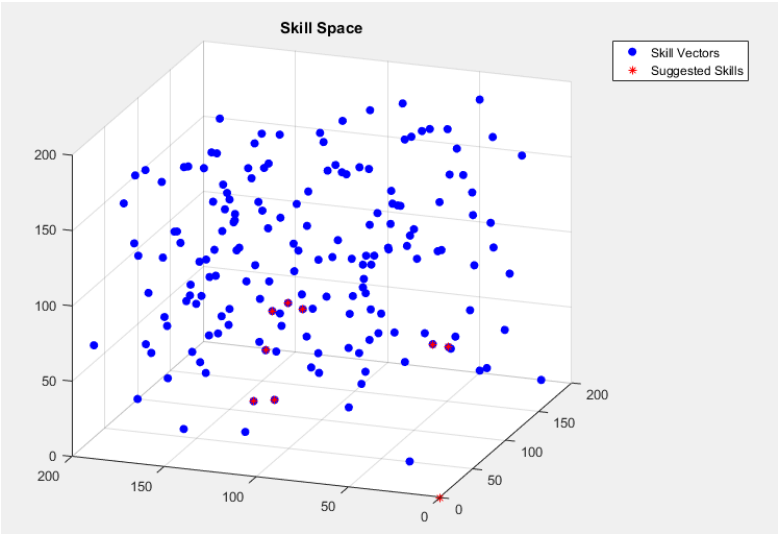

Fig. 3: Scatter Plot for Module 1

```
Suggested skills for user 1:
    'Manual'

    'Java'

Suggested skills for user 2:
    'TestManagement'

    'BluetoothTesting'

Suggested skills for user 3:
    'Manual'

    'Java'

Suggested skills for user 4:
    'WebDevelopment'

    'ElectricallytrainedAutomationEngineer'
```

Fig. 4: Matlab Output for Module 1

```
For
Software Developer
    'J2EE'

    'Assurance'

    'EducationalMarketing'

For
Automation Engineer
    'ECS'

    'QualityEngineeringAutomation'

    'WebDevelopment'

    'J2EE'

    'BusinessAnalysis'

    'Assurance'

    'EducationalMarketing'

    'SupplyChainManagement'

    'SSIS'
```

Fig. 5: Matlab Output for Module 2

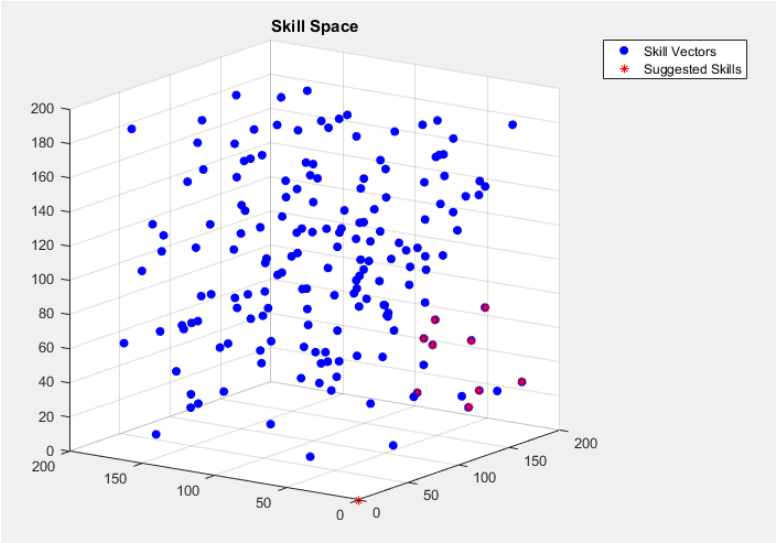

Fig. 6: Scatter Plot for Module 2

VI. Conclusion

The proposed method recommends a career path in terms of skill set based on current skills of candidate and also keeping in view the candidate's career goal. Although the proposed method is successful, it faces some challenges. Following are some challenges.

i) There is some difficulty in choosing the threshold value for skill recommendation in algorithm 3.

ii) This algorithm does not take into account the experience of the candidate including the previous job position.

iii) Although the algorithm coverges the parameters to desired output, the complexity also depends upon the inital random values chosen.

References

[1] https://en.wikipedia.org/wiki/Vector_space

[2] https://www.desmos.com/calculator

[3] https://en.wikipedia.org/wiki/Collaborative_filtering

[4] https://en.wikipedia.org/wiki/Transformation_matrix