Career Navigator: A Recommender Systems Approach

Shah Parth Viren(1401054)

AOBD-Final Exam
School of Engineering and Applied Science, AU

***Abstract*—This paper proposes a method for facilitating users to navigate their careers. User can either enter his/her profile based on which different skills for varied careers are suggested. User can also enter the career goal he/she wishes to achieve, and the system recommends a skill-set for achieving this goal. A recommender system approach is used here where based on the given *feature vector*, our system tries to predict the *parameter vector* of a particular career.As paper address two different problems, one of suggesting various extra skills for opting different career path, second of suggesting a skill-set for particular career path,two modules are proposed in this paper.**

Keywords: Recommender Systems, Feature finders, Stochastic gradient descents.

I. Introduction

Problem of finding proper skills for a desired career or getting skill-sets required for different careers, from a given user's incomplete profile is discussed here. Comprehensive research on this problem using SVM, K-means clustering, Fused Laso, is proposed in [1]. In this paper I like to introduce a recommender system which recommends, skill-set to user using collaborative filtering. Business social networking gaint LinkedIn uses *BrowseMaps*, for recommending connections, these *BrowseMaps* also uses the concepts of collaborative filtering at the back end[2].In our case instead of a complete collaborative filtering where both *feature vector* and *parameter vector* are simulataneously updated, a simple recommender system is used.As *feature vector* is already known we just need to iteratively find the *parameter vector*. The paper suggests a method where *feature vectors* are provided and*parameter vectors* update themselves , and using these two parameters we try to solve the above mentioned problems.

II. Recommender Systems

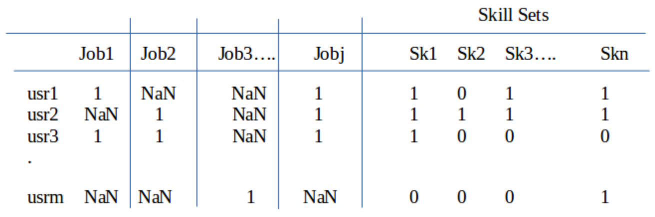
In this section I would like to discuss the algorithm for recommender systems in general. Recommender systems learns the *parameter vectors* from the given *feature vector*.

feature vectors: The features (in our case skills) a particular parameter (in our case candidate) possesses is called feature vector for that parameter. If a candidate has n skills then (nx1) would be the dimension of feature vector. For any arbitary candidate i feature vector would be $x^{(i)}$.

parameter vectors: The parameter vector obtained for a set of variables (in our case different career) is called parameter vector. If the dimension of $x^{(i)}$ is (nx1) then parameter vector for a particular variable would be also (nx1). Parameter vector for any arbitary career (*Software Engineer*) j would be $\theta^{(j)}$.

A. *Algorithm*

Algorithm here is explained using the problem scenario itself, so the variables defined and used would carry the same meaning in section III.



					Skill Sets			
	Job1	Job2	Job3….	Jobj	Sk1	Sk2	Sk3….	Skn
usr1	1	NaN	NaN	1	1	0	1	1
usr2	NaN	1	NaN	1	1	1	1	1
usr3	1	1	NaN	1	1	0	0	0
.								
usrm	NaN	NaN	1	NaN	0	0	0	1

Table-1: Sample scenario

In the above given table *usr* is the candidate having *feature vector* $x^{(i)}$. As we already have feature vector we need to find the parameter vector $\theta^{(j)}$ for different careers *Job1, Job2, Job3,...*

Variables for the algorithm is defined as follows

n_j = number of career options

n_u = number of candidates

$r(i,j)$ = 1 if candidate i has applied for a job in particular career j

$y(i,j)$ = 1 (defined 1 only if $r(i,j)$ = 1)

$x^{(i)}$ = feature vector for candidate i

$\theta^{(j)}$ = parameter vector for career option j

Using the above parameter the cost function can be defined as[3]:

To learn $\theta^{(j)}$,

$$min_{\theta^{(j)}} = \frac{1}{2}\sum_{i:r(i,j)=1}((\theta^{(j)})^T(x^{(i)})-y^{(i,j)})^2+\frac{\lambda}{2}\sum_{k=1}^{n}(\theta_k^{(j)})^2 \tag{1}$$

The above equation is similar to a simple linear regression. The base of the summation is choosing all i for which $r(i,j) = 1$ i.e. choosing only the candidates who have $y(i,j) = 1$, for a particular career j.

Parameter vectors θ for all career path $(job1, job2, ...)$, can be calculated using the following cost function $J_{min(\theta^{(1)},\theta^{(2)}....,\theta^{(n)})}$.

$$min_{\theta^{(1)},..\theta^{(n)}} = \frac{1}{2}\sum_{j=1}^{n_j}\sum_{i:r(i,j)=1}((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)})^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(\theta_k^{(j)})^2 \quad (2)$$

Let α be the learning rate, then the parameters are updated as,

$for(j = 1 \ to \ n_j)$

$$\theta_k^{(j)} = \theta_k^{(j)} - \alpha\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right) \quad (3)$$

Using the above given equation a simple algorithm for recommender systems can be given as

```
(1) Initialize parameter vectors $\theta$ with some random values.
(2) Using equation (2) find cost function $J_{min(\theta^{(1)},\theta^{(2)}....,\theta^{(n)})}$.
(3) Update the value of $\theta$ using (3).
```

Algorithm-I

This process is repeated till cost function is less than a particular δ.If this δ is achieved after P iterations and as there is already a loop of size of parameter vector $(n \times 1)$ in each iteration for updating θ , time complexity of this algorithm would be $O(Pn)$

III. Designing Modules

This section discusses the modules we need to design for solving the two given problems. Module-I answers to the question of suggesting different skills to the candidate based on his/her profile. Module-II suggests skill-sets particular to a career.Variables are to be refered from section - II

While designing this modules the top 25 skills of entire given dataset[4] were considered for *feature vectors*, skills different than these skills were marked as others.

Thus $x^{(i)}$ is (26×1), $\theta^{(j)}$ is also (26×1). There are 39 different career options given in dataset hence, $n_j = 39$. There are 784 candidate entries, hence $n_u = 784$.

A. *Module-I*

Using the above mentioned Algorithm-I, we now already have our parameter vector θ.

From table-I, we see that, we have many NaN, values.Hence if a candidate i has a NaN for career j, then $y^{(i,j)}$ for this NaN, can be calculated by

$$y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$$

This $y^{(i,j)}$ would be surely $0 \leqslant y^{(i,j)} \leqslant 1$

Let us define a threshold Ω.

We define a candidate i, suitable for career j, only if $y^{(i,j)} \geqslant \Omega$

Thus if $y^{(i,j)} \leqslant \Omega$, this means that candidate i lacks certain skills for career j.

Thus now the *feature vector* $x^{(i)}$ needs to be updated, this update is easy as we already have $\theta^{(j)}$.

$\theta^{(j)}$ are the weights of skills required for a particular career. Thus we just need to suggest the rows in $x^{(i)}$ which are zero (mising skills), but have weight in $\theta^{(j)}$.

```
Calculate $\theta$ from Algorithm-I, and take $\theta^{(j)}$.
For a candidate $i$, use feature vector $x^{(i)}$
for($\forall\ j = NaN,\ for$ candidate $i$)
  Calculate $y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$
  if($y^{(i,j)} \leqslant \Omega$)
    print $\forall\ k\ such\ that(x(k)^{(i)} = 0\ and\ \theta(k)^j \neq 0)$
```

Algorithm-Module I

After getting parameter vector θ, a loop for all s occurrences of NaNs of candidate i, is iterated hence the time complexity would be $O(s)$, this time complexity is taken by assuming that θ is already available.

B. *Module-II*

Designing this module is pretty simple as we already have a career goal i.e. $y^{(i,j)}$ for candidate i and career j, we just need to suggest all the weights of $\theta^{(j)}$ for that particular career choice where $x^{(i)} = 0$ and our job is done.

```
Candidate $i$ enters his/her career goal$j$, $y^{(i,j)}$.
From algorithm-I we have parameter vector $\theta$.
print $\forall\ k,\ such\ that\ (x^{(i)}(k) = 0\ and\ \theta^{(j)}(k) \neq 0)$
```

Algorithm-Module II

After getting parameter vector θ, this Module takes just $O(1)$ time complexity.

IV. Generated Outputs

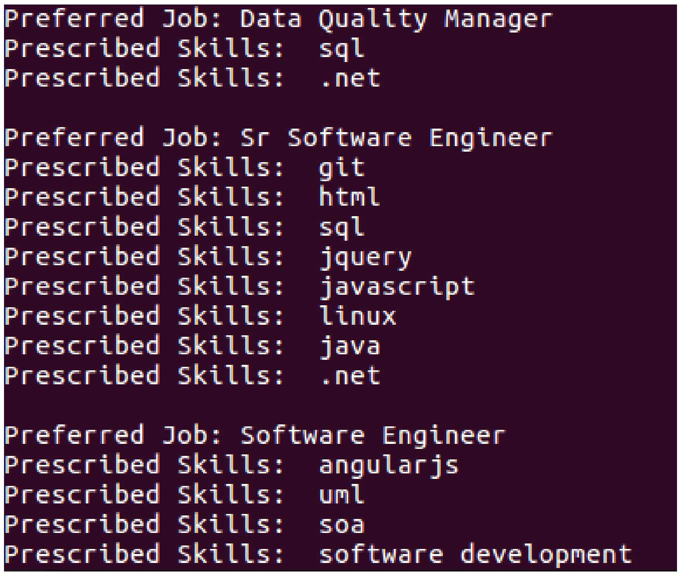
Implementing the above module for following data, which is taken from dataset mentioned in [4].

Candidate id (i) = 100;

Interested Job (j) by candidate i = Technical Support Specialist ;

A. *Module-I*

This module gives the output just by reading Candidate profile for (i = 2)

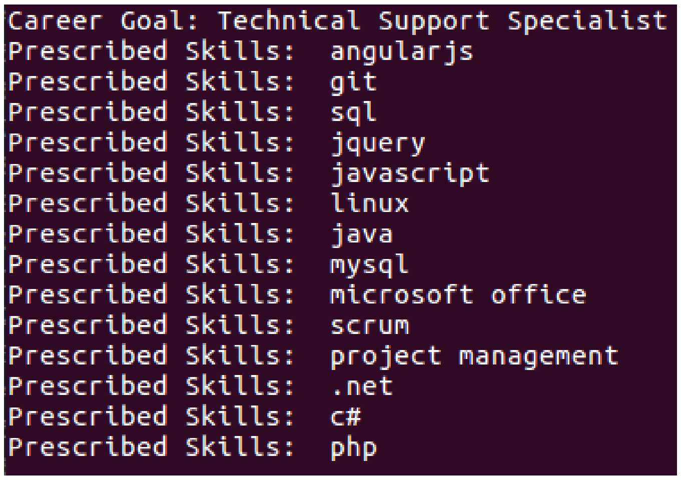


```
Preferred Job: Data Quality Manager
Prescribed Skills:  sql
Prescribed Skills:  .net

Preferred Job: Sr Software Engineer
Prescribed Skills:  git
Prescribed Skills:  html
Prescribed Skills:  sql
Prescribed Skills:  jquery
Prescribed Skills:  javascript
Prescribed Skills:  linux
Prescribed Skills:  java
Prescribed Skills:  .net

Preferred Job: Software Engineer
Prescribed Skills:  angularjs
Prescribed Skills:  uml
Prescribed Skills:  soa
Prescribed Skills:  software development
```

Fig-1: Module-I Output

Fig-1 shows the career options for Candidate with $i = \mathbf{100}$ and suggests additional skills to be acquired skill-set.

B. *Module-II*



```
Career Goal: Technical Support Specialist
Prescribed Skills:  angularjs
Prescribed Skills:  git
Prescribed Skills:  sql
Prescribed Skills:  jquery
Prescribed Skills:  javascript
Prescribed Skills:  linux
Prescribed Skills:  java
Prescribed Skills:  mysql
Prescribed Skills:  microsoft office
Prescribed Skills:  scrum
Prescribed Skills:  project management
Prescribed Skills:  .net
Prescribed Skills:  c#
Prescribed Skills:  php
```

Fig-2: Module-II Output

Fig-2 shows the skills required for getting a job as Technical Support Specialist.

V. Conclusions

In this paper, recommender systems approach is used for getting the parameter vector $\boldsymbol{\theta}$. This parameter vector is now used in module-I and module-II for geting the desired output. In terms of candidate's profile we just have considered skills, and no other parameter was taken into consideration. This can be further extended by looking at the approach opted by [2] for looking at various parameters present is a candidate's profile.

References

[1] Fortune Teller : Predicting Your Career Path. Ye Liu, Luming Zhang, Liqiang Nie, Yan Yan, David S. Rosemblum.

[2] The Browsemaps: Collaborative Filtering at LinkedIn.Lili Wu, Sam Shah, Sean Choi, Mitul Tiwari, Christian Posse

[3] Recommender Systems:An introduction to Machine Learning (Week-9, video Lectures), Andrew Ng.

[4] DataSet available on `Gyapak\Acedemic \Ratnik Gandhi \Candidate Profile Data`.