Career Job Finder using Recommender Systems

Bhavya Patwa (1401063)

AOBD - Final Exam, April 2017
School of Engineering and Applied Science, Ahmedabad University

***Abstract*—The goal of this paper is to provide career solutions to its users seeking for bettering their career path. This will help users build up careers and improving the skillset in order to achieve a career goal. These career solutions, in terms of skillset, would be suggested by a Recommender System, trained from the existing skills and job preferences of users. The incomplete data of the remaining job preferences of users would be filled by the modules designed by using the trained data on the Recommender System. Module 1 tackles the generation of skillsets recommendation for best matched jobs, while Module 2 would provide skillsets to be required for a particular career goal as per by the user.**

Keywords: Recommender Systems, Stochastic Gradient Descent, Job Finder

I. Introduction

The approach towards the huge logistics problem faced by the majority of the job hunters is addressed in this paper. Users that are opting to change their careers are often unaware of what additional skills do they require to learn. This is where the Recommender Systems dive in to the rescue. Based on skills and job preferences of all the users simultaneously, the Recommender System would pitch new job careers that a particular user might be willing to opt for, and the required skills that the user is missing in his/her profile. Business Social Networking giants such as LinkedIN uses *The BrowseMaps* for recommending connections, which is based on a *Collaborative Filtering* approach which again is derived from the concepts of Recommender Systems[1].

II. Data cleaning and parsing

The JSON data provided was of the candidates that have shown interest in particular jobs which are there in form of the file names. However, it would have resulted into several anomalies when it came to actual processing on the data into our model i.e. unicode symbols, duplicate entries in other job interests, extra white spaces, no. of years in every skill etc. So many tasks were ran as a form of regex queries, data modification on assigning unique candidate IDs, merging IDs duplicate candidate profiles, pruning specific skillsets based on count of occurrence etc. Thus, cleaning and preprocessing on the data was done as a necessity in order to fit the model described in this paper.

III. Recommender Systems

In this section I would like to discuss the algorithm for Recommender Systems in general. Recommender Systems learns the *parameter vectors* $\theta^{(j)}$ from the *feature vectors* $x^{(i)}$ and the *observed data* $y^{(i,j)}$.

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 (n $\times$ 1) would be the dimension of a single feature vector. For any arbitary candidate, the i^{th} feature vector is denoted as $x^{(i)}$.

Parameter Vectors: The parameter vectors obtained for a set of variables (in our case $x^{(i)}$ and $y^{(i,j)}$) is called parameter vector $\theta^{(j)}$. If the dimension of $x^{(i)}$ is (n$\times$1) then parameter vector for a particular variable would be also (n$\times$1). Parameter vector for any arbitary career (*Test Engineer*) j would be $\theta^{(j)}$.

A. *Algorithm*

Algorithm here is explained for the problem addressed, so the variables which are defined and used would carry the same notation in the further sections.

*1) **Model Description**:* From the given dataset, n_u is the number of candidates, each having skills in form of *feature vector* $x^{(i)}$. As we already have a *feature vector* we need to find a parameter vector $\theta^{(j)}$ for a different career n_j.

The observation variables are defined as:

n_j = number of career options
n_u = number of candidates
$r(i,j)$ = 1 if candidate i has applied for a job in career j
$y(i,j)$ = 1 (defined only if $r(i,j)$ = 1)
$x^{(i)}$ = feature vector for candidate i
$\theta^{(j)}$ = parameter vector for career option j
$\Theta = [\theta^{(1)}, \theta^{(2)}...., \theta^{(n)}]$

Using the above given parameters, the cost function can be defined as[24]:

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

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

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

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

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

IV. Designing Modules

Two modules are designed which provide suggestions for skill-set along with the career-path to the user and just skill-set for a given career-path desired by the user.

While designing this modules the top 25 skills of entire given dataset[3] 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 $n_u = 784$.

						SKILLS			
	JOB1	JOB2	JOB 3	…..	JOB k	Skill 1	Skill 2	……	Skill n
Candidate 1	1	1	1		NaN	1	1	……	1
Candidate 2	1	1	NaN		NaN	1	0	……	1
:									:
:									:
Candidate j-1	NaN	NaN	1		NaN	0	1	……	0
Candidate j	1	NaN	NaN		1	0	1	……	1

Table-1: Sample scenario

A. Module-I

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

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

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

where, $0 \leqslant y^{(i,j)} \leqslant 1$

Let us define a threshold α for predicting if a user belongs to a certain job career.

We define a candidate i, suitable for career j, only if the $y^{(i,j)} \geqslant \alpha$. Thus, if $y^{(i,j)} \leqslant \alpha$, this means that candidate i lacks certain skills for career j.

Thus, now we get the predicted *feature vector* $x^{(i)}$ for a user i, as we already have the $\theta^{(j)}$ in form of weights of the *feature vectors*.

The new required skills can now be easily found as we just need to find the k rows of feature vector $x^{(i)}$ $(n \times 1)$ which have value 0, but have positive weights in $\theta^{(j)}$. The index of the zero filled cells would represent a skill user requires along with his/her current skills if he/she wants a career in that particular j.

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.

V. Time-Complexities

A. Algorithm-I

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

B. Module-I

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

C. Module-II

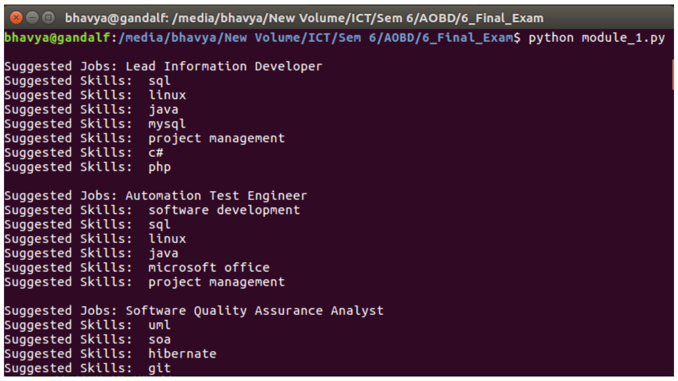
After getting parameter vector Θ, this Module takes just $O(1)$ time complexity since the output generated would be a product of 1×n and n×1 matrix.

VI. Results

Implementing the above module for following data, which is taken from dataset mentioned in [3].

Candidate id (i) = 1;

A. Module-I

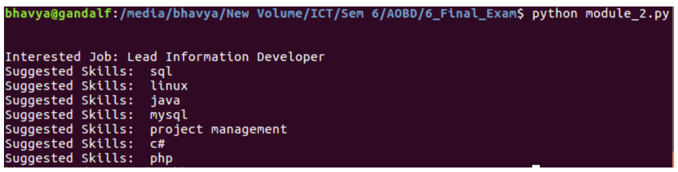


```
bhavya@gandalf: /media/bhavya/New Volume/ICT/Sem 6/AOBD/6_Final_Exam
bhavya@gandalf:/media/bhavya/New Volume/ICT/Sem 6/AOBD/6_Final_Exam$ python module_1.py

Suggested Jobs: Lead Information Developer
Suggested Skills:  sql
Suggested Skills:  linux
Suggested Skills:  java
Suggested Skills:  mysql
Suggested Skills:  project management
Suggested Skills:  c#
Suggested Skills:  php

Suggested Jobs: Automation Test Engineer
Suggested Skills:  software development
Suggested Skills:  sql
Suggested Skills:  linux
Suggested Skills:  java
Suggested Skills:  microsoft office
Suggested Skills:  project management

Suggested Jobs: Software Quality Assurance Analyst
Suggested Skills:  uml
Suggested Skills:  soa
Suggested Skills:  hibernate
Suggested Skills:  git
```

Fig. 1. Suggestions of jobs with required skills by Module - I

B. Module-II

Interested Job (j) = Lead Information Developer



```
bhavya@gandalf:/media/bhavya/New Volume/ICT/Sem 6/AOBD/6_Final_Exam$ python module_2.py

Interested Job: Lead Information Developer
Suggested Skills:  sql
Suggested Skills:  linux
Suggested Skills:  java
Suggested Skills:  mysql
Suggested Skills:  project management
Suggested Skills:  c#
Suggested Skills:  php
```

Fig. 2. Suggested skills of career goal by Module - II

VII. Conclusion

The approach of suggesting career-path to users using Recommender Sytems is achieved by training our parameter vectors $\boldsymbol{\theta^j}$. These $\boldsymbol{\theta^j}$s were used in Module-I and II for obtaining desired outputs. We have considered skills as a feature vector parameter from a candidate's profile, which can be further extended by using more parameters to make our model truly diverse[2].

References

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

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

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