Final Exam - Recommendation System

Aashima Yuthika - 1401071
School of Engineering and Applied Sciences, Ahmedabad University
Subject: Algorithms and Optimisation for Big Data
April 28, 2017

I. Introduction

A Recommendation System is an information filtering system that suggests certain paths that a user can take depending on what path previous users have taken. These systems have been incredibly popular in the recent years. They are widely used by e-commerce sites, by sites like LinkedIn that give recommendation to users for possible connections that they can add to their circles. The things that one needs to keep in mind while designing a recommendation system are as follows:

- **Quality of Predictions:** The predictions should not be random, and should consider criteria that will help it make more relevant predictions.
- **Speed:** Most recommender systems work in a commercial and/or online setting, and so it is important that they can start making recommendations for a user almost instantly.
- **Scalability:** Systems in a commercial and/or online setting can have a huge dataset. The algorithm must maintain its speed even if there is a huge amount of data.
- **Easily Updated:** The datasets behind recommender systems are constantly being updated. So, the Algorithm should be able to handle this.
- **Sparse Data Handling:** Sometimes when the datasets are very sparse, we still want to be able to make good predictions.

Discussed in the next section are some common algorithms used in recommender systems.

II. Some Common Algorithms

A. Collaborative Filtering

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. 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, like the k-nearest neighbor (kNN) approach. This approach may suffer from scalability and sparsity problems.

B. Content Based Filtering

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the users preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are similar to those that a user liked in the past (or is examining in the present). In particular, various candidate items are compared with items previously rated by the user and the best-matching items are recommended. A key issue with content-based filtering is whether the system is able to learn user preferences from users' actions regarding one content source and use them across other content types.

C. Hybrid Recommender Systems

Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways:

- By making content-based and collaborative-based predictions separately and then combining them
- By adding content-based capabilities to a collaborative-based approach (and vice versa)
- By unifying the approaches into one model

III. My Approach

All the codes, along with the README and this report have been uploaded to the GitHub repository - 'https://github.com/AashimaYuthika/AOBD17_1401071' in the 'submission5' folder.

A. Data Cleaning

The following pre processing was done on the data before actually using it for recommendation:

- The data contained many non-ASCII characters like bullet points, right arrow and other garbage value. This was

removed. This was done before the data was used in the recommender modules.
- The separation of skills in the data were based on different delimiters. Some were separated by '&&' while some had ',' separating them. This was made uniform and all these were separated using commas. This was done while processing the data for the recommender system.
- There were duration for which a user has had that skill. This was removed as there was no uniformity in it.

B. Recommendation

For the recommendation system I have used some ideas from Collaborative Filtering. There are basically two approaches that I have tried for the same.

1) Approach 1

- This first sees what career goal is right for the user or what the user wants
- It then makes an exhaustive list of all the skills that users have in that profession
- We then make a matrix S of each user and their skills for that profession
- The matrix S is then summed along the column to find how many users have a particular skill
- Then the top 3 frequencies are taken
- All skills having the above frequency are shown to the user

2) Approach 2

- This first sees what career goal is right for the user or what the user wants
- It then makes an exhaustive list of all the skills that users have in that profession
- We then make a matrix S of each user and their skills for that profession
- We then make a matrix T of pairs of skills and how many time they occur together
- 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:

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

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

IV. Results

A. Module 1

The Career is selected randomly and the user to whom career path is to be suggested is too. Below is the result using the Frequency of Skills Approach (Approach 1) and the Jaccard Index Approach (Approach 2).

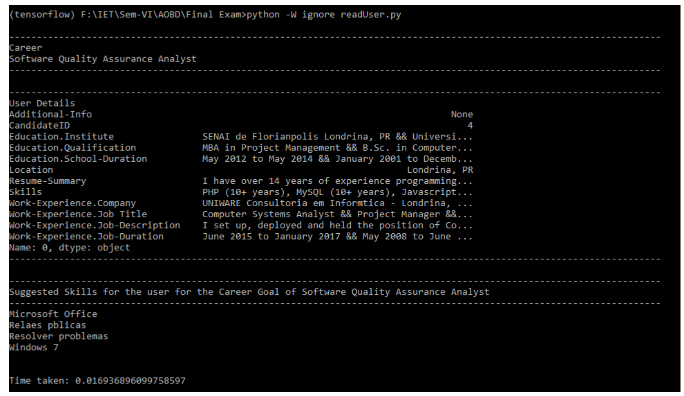

Fig. 1: Suggesting Career Path using Frequency Approach. Execution Time: 0.017 seconds

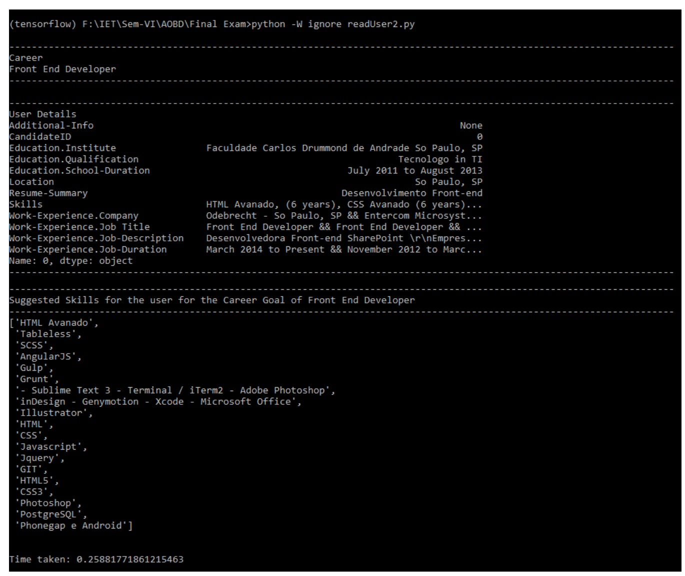

Fig. 2: Suggesting Career Path using Jaccard Indices Approach. Execution Time: 0.259 seconds

B. Module 2

The User is asked to select a Career from the list of options and then the suggested skill set is shown to the user. Below is the result using the Frequency of Skills Approach (Approach 1) and the Jaccard Index Approach (Approach 2).

```
(tensorflow) F:\IET\Sem-VI\AOBD\Final Exam>python -W ignore userGoal.py
1) Automation Test Engineer
2) Computer Systems Manager
3) Customer Support Administrator
4) Customer Support Specialist
5) Data Center Support Specialist
6) Data Quality Manager
7) Database Administrator
8) Desktop Support Manager
9) Desktop Support Specialist
10) Front End Developer
11) Java Developer
12) Junior Software Engineer
13) Lead Information Developer
14) Senior IT Architect
15) Senior Network Engineer
16) Senior Network System Administrator
17) Senior Programmer Analyst
18) Senior Security Specialist
19) Senior Software Engineer
20) Senior System Architect
21) Senior System Designer
22) Senior Systems Analyst
23) Senior Web Administrator
24) Senior Web Developer
25) Software Architect
26) Software Developer - Backend
27) Software Developer
28) Software Engineer
29) Software Quality Assurance Analyst
30) Sr. Software Engineer
31) support engineer
32) System Architect
33) Systems Analyst
34) Systems Designer
35) Technical Operations Officer
36) Technical Specialist
37) Technical Support Specialist
38) Telecommunications Specialist
39) UI Developer
Choose your Career Goal from the options above >> 10
```

Fig. 3: Giving options to the user for Career Path

```
--------------------------------------------------------------------------------
You chose
Front End Developer

--------------------------------------------------------------------------------
Suggested Skills for the user for the Career Goal of Front End Developer
--------------------------------------------------------------------------------
HTML
CSS
Photoshop
Jquery
GIT


Time taken: 0.026953381413717236
```

Fig. 4: Suggesting Career Path using Frequency Approach. Execution Time: 0.027 seconds

```
--------------------------------------------------------------------------------
You chose
Front End Developer

--------------------------------------------------------------------------------
Suggested Skills for the user for the Career Goal of Front End Developer
--------------------------------------------------------------------------------
['HTML Avanado',
 'Tableless',
 'SCSS',
 'AngularJS',
 'Gulp',
 'Grunt',
 '- Sublime Text 3 - Terminal / iTerm2 - Adobe Photoshop',
 'inDesign - Genymotion - Xcode - Microsoft Office',
 'Illustrator',
 'HTML',
 'CSS',
 'Javascript',
 'Jquery',
 'GIT',
 'HTML5',
 'CSS3',
 'Photoshop',
 'PostgreSQL',
 'Phonegap e Android']


Time taken: 0.39154865733425415
```

Fig. 5: Suggesting Career Path using Jaccard Indices Approach. Execution Time: 0.392 seconds

The Frequency Approach, although has a time complexity $O(n^2)$ it's suggestions may not be as beneficial as that of the Jaccard Indices Method, which has a time complexity of $O(mn^2)$ where m is the number of users and n is the number of skills. This is true because the second method takes into account as to which skills are most frequently seen together and hence, giving a better and more connected career path to the user.

References

[1] http://stackoverflow.com/
[2] https://www.codementor.io/jadianes/build-data-products-django-machine-learning-clustering-user-preferences-du107s5mk
[3] http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/algorithms.html
[4] https://www.quora.com/Which-algorithms-are-used-in-recommender-systems
[5] https://www.dataquest.io/blog/pandas-python-tutorial/
[6] https://en.wikipedia.org/wiki/Collaborative_filtering
[7] https://en.wikipedia.org/wiki/Recommender_system
[8] http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note02-2up.pdf
[9] https://en.wikipedia.org/wiki/Jaccard_index