Analysis And Optimization of Big Data Final Exam Report

Malav Vora
1401062

***Abstract*—Abstract In the given problem, we are given a big data of various details about many candidates. Based on this data, we apply big data algorithms and suggest a career path for the candidate based on his profile and skillset he possesses.**

I. Task - 1: Cleaning of data

The given data is corrupted with some unicode values in it which are required to be removed/replaced with proper replacement. So, first we need to detect the unicodes by reading complete files and then replace it with proper content such that it is easier for us to read data when we use it for algorithms. First of all we read whole file and detect where the unicodes are. Whenever a unicode is detected, depending on what the unicode represents, the python code replaces it with its equivalent ASCII code so that is becomes easier for us to read it. This goes on till the end of file and then we have a data that is cleaned and completely in ASCII characters. Then this data is written in the file and algorithm are applied on it.

II. Task - 2 : Structuring of data

To structure the clean data, it is converted to .csv from .txt file because it data in a .csv file is more structured and processing is easily possible in it than a .txt file. So, using json library of python, I load the whole file and seperate its content according to the fields provided like CandidateID, Company, job-Description, etc. This helps us get all the details of a candidate in a single column. This also stores all the data in python dictionary because of the format of given data. The algorithm for it is as follows:

```
# loads .txt file using json to read
with open("Automation Test Engineer
    .txt", "rb") as fin:
    content = json.load(fin)

# dumps content of .txt file using json
# in the new file
with open("Automation Test Engineer json
    .txt", "wb") as fout:
    json.dump(content, fout, indent=1)

# load data of new json file using in
# dictionary x
with open("Automation Test Engineer json
    .txt", "rb") as fin:
        x = json.load(fin)

f = csv.writer(open("filename.csv", "wb+"))

# Write CSV Header
f.writerow(["CandidateID", "Company",
            "Job-Description", "Job Title",
            "Job-Duration", "Skills",
            "Additional-Info", "Location",
            "Institute", "School-Duration",
            "Qualification", "Resume-Summary"])

for x in x:
    f.writerow([x["CandidateID"],
                x["Work-Experience"]["Company"],
                x["Work-Experience"]["Job-Description"],
                x["Work-Experience"]["Job Title"],
                x["Work-Experience"]["Job-Duration"],
                x["Skills"],
                x["Additional-Info"],
                x["Location"],
                x["Education"]["Institute"],
                x["Education"]["School-Duration"],
                x["Education"]["Qualification"],
                x["Resume-Summary"]])
```

III. Module 1: Read user's profile and suggest a career path - based on his skill set

To suggest the candidate about his career path, we use the concept of nearest neighbours where more the common skills between candidates, nearer are the neighbours and hence more is their probability of having same career path. This methodology is also known as collaborative filtering which is used for recommending a user based on rating or reviews of other users.

Now that we have organised data with us, we need to suggest a career path for a candidate. As the candidate logs in, we have his CandidateID and all of his details. We need to provide our suggestions based on the skills a candidate possesses. So, I take our candidate for whom we are suggesting a career path and all the other candidates with same profile(i.e. Automation Test Engineer, Computer Systems Manager, Customer Support Administrator, etc.) and check for the common skills between them. Then a list is created which has the candidates who have one or more skills common with the logged in candidate. Also this list is sorted such that other candidates having highest number of common skillset is first because it implies that both of them have more common skills and so there is higher probability that the current candidate will follow his career path. Once we get the candidates whose path the current candidate is supposed to follow, we then take job detail of those candidates and suggest it to our candidate. The module that gives similarities between the skillset of a candidate and other candidates is given below. It returns a list of his skillset with another candidate.

```
def candSimilarity(candidate1, candidate2):
"""Computes the similarity between two
candidates and their skills. Both
candidate1 and candidate2 are
dictionaries. Candidate1 is the candidate
whom we are suggesting career path and
candidate2 will be all other candidates
having same title"""
  similarity = []
  for skill in candidate1:
    if skill in candidate2:
          similarity.append(candidate1[skill])
  return similarity
```

This method gives the closest candidates i.e candidates having most common skillset with the candidate. It is a 2-dimensional list that has list of skills in common with every other candidate.

```
def closestCandidates(candidate1,candidates):
"""creates a sorted list of users based
on skills to username"""
 similarity = []
 for candidate in candidates:
   if candidate != candidates:
    similarity = candSimilarity(candidate1,
                                candidate)
    similarity.append(similarity, candidate1)
    # sort based on distance -- closest first
   similarity.sort()
 return similarity
```

The method that helps a candidate find relatable career path is given below. It first checks the nearest neighbours i.e. candidates having most common skillset. Then using that it suggests the career path to the candidate depending on what other candidates have had already but this candidate hasn't.

```
def suggestCareerPathSkillset(candidate,
                              candidates):
"""Give list of suggestions"""
# first find nearest neighbor
 closest = closestCandidates(candidate,
                             candidates)
 suggestions = []
 closeCandPath = candidates[jobTitle]
 candidateJob = candidate[jobTitle]
 for jobTitle in closeCandPath:
  if not jobTitle in candidate:
   suggestions.append(job-title,
                      closeCandPath[jobTitle])
 return sorted(suggestions,
               skillset: skillset[1])
```

IV. Module 2: User enters a career goal and suggest a career path - based on his career goal

In this module we take a career goal as an input from user. Then we need to provide him a career path based on his goal. Here also we will have to use nearest neighbour and collaborative filtering. But here the nearest neighbours with whom we will compare will be only those candidates who have already achieved the career goal entered by user. We form a cluster of candidates who have achieved the goal and that is in their career path. Then using the concept of nearest neighbours, we find out candidates who have job profile and other additional information closest to the user. We sort them in order such that the nearest one is closest and so. From those candidates we select the path chosen by them from the user's current title to his goal. It is not necessary that each of those candidate will have same path. So we give the path in order that user needs to acquire minimum skill-set. In this way, user will then only be suggested career path based on his career goal. Fig. 2 is a flowchart explaining the flow from user entering his goal to the module suggesting him his career path.

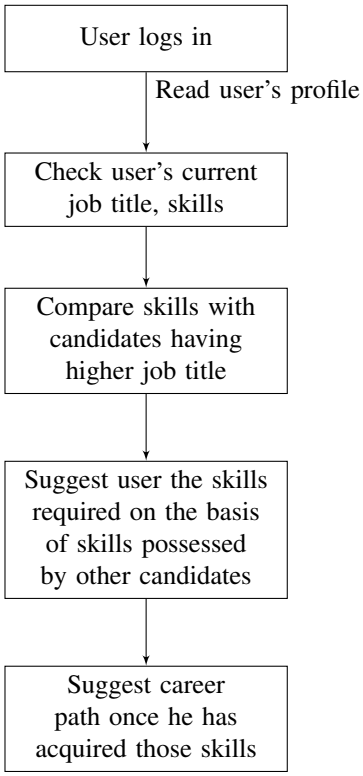

Fig. 1. Flowchart of how user will be suggested his career path

V. Results

Figure 3 and figure 4 are the results for Candidates 4 and 7 of Automation Test Engineer respectively.

VI. Conclusion

I was able to clean the data and fetch ASCII data such that processing is possible on that data. Then I applied collaborative filtering algorithm for designing first module. I also tried to design second module but there were issues of reading other fields that had descriptive responses.

References

[1] JohnS. Breese, David Heckerman and Carl Kadie . *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. Microsoft Research Redmond, WA 98052-6399

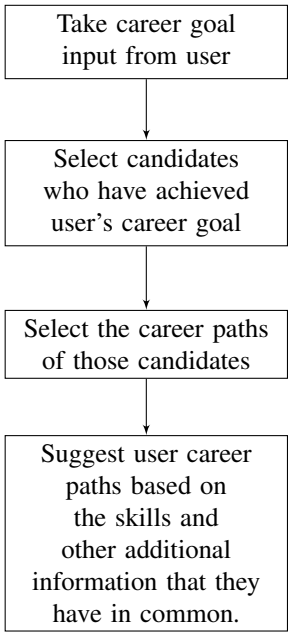

Fig. 2. Flowchart of how user will be suggested his career path on the basis of career goal

Fig. 3. Career Path for Candidate 4 of Automation Test Engineer

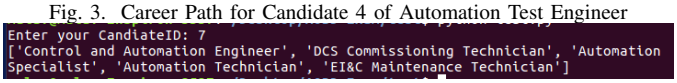

Fig. 4. Career Path for Candidate 7 of Automation Test Engineer

[2] Ron Zacharski *A Programmer's Guide to Data Mining: The Ancient Art of the Numerati*.

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl *Item-Based Collaborative Filtering Recommendation Algorithms*. GroupLens Research Group/Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455

[4] Python Documents, `https://docs.python.org/3/library/json.html`

[5] Stackoverflow, `http://stackoverflow.com/questions/3368969/find-string-between-two-substrings`

[6] json, `http://stackoverflow.com/questions/20901018/converting-string-file-into-json-format-file`

[7] Substring, `http://stackoverflow.com/questions/12572362/get-a-string-after-a-specific-substring`

[8] Unicode to ASCII, `http://stackoverflow.com/questions/175240/how-do-i-convert-a-files-format-from-unicode-to-ascii-using-python`