Final Project Report on

Job Recommendation Engine

By

Vijay Prayagala

International School of Engineering

Batch14 -2016

Table Contents

1. Overview 3
2. Purpose 4
3. State of Art Recommenders 4
   1. Approach and Techniques 4
4. Method - Hybrid Approach 7
   1. Problems with Data 7
   2. Common Problems with Recommenders 8
   3. Evaluation of Recommenders 8
   4. Steps followed to build recommender using SVD 9
5. Data Gathering and Understanding the Data 10
   1. Data Gathering 10
   2. Understanding the Data 12
6. Results 15
7. Analysis 16
8. Appendices 16

# Overview

Quickhire is a US based job portal which is fairly new in the market for job seekers and has data collected for 2014-15 year. The portal is becoming quite popular among seekers for part-time jobs.

They have collected all the data related to jobs and candidates in their portal. Basic information about jobs, company and candidates are recorded during registration and job posting process along with some detailed information regarding applicant interaction. The data includes job views which deals with time spent by the candidate on every job view and if they have applied for the job or not, etc.

A simple job recommendation system has been implemented by Quickhire to bring a list of jobs that are likely to be preferred by the candidates. They are currently shortlisting the jobs based on two relevance factors such as distance between the job and candidate’s location and type of industry to make recommendations to the candidates.

# Purpose

The aim of the current project is to build another job recommendation using collaborative filtering which recommends jobs based on collaboration between Applicants, Jobs. The similarities are extracted from rating data that the candidates give to the jobs.

Reviews on the job details and candidate profiles are also taken to make more appropriate recommendations. Information’s that are useful to improve recommendation such as profiles, location, education, industry, etc. are examined and incorporated in the model of this job recommendation engine. The recommender is expected to improve the quality of job matching process implemented in Quickshire.

# State of Art Recommenders

There are several techniques and approaches that have been known for building good recommendation engines. These recommendation systems are aimed to rate the jobs based on similarity measures. It is at high level is kind of predicting the rating for item which is not known to user.

However, different users may have different characteristics and a single recommendation approach may not be suitable for all users. Therefore, a job recommendation system should have the ability of using other available data to recommend items that are appropriate to users.

# Approach and Techniques

We we can classify jobs recommendation approaches into two categories:

1. **Content-Based Approach**

The principle of a content-based recommendation is to suggest items that have similar content information to the corresponding applicants. The personal information and their job desires including the job description posted by recruiters and the background description of enterprises, are used as the content for recommendation.

The basic process of content-based recommendation is acquiring the content information of job candidates and jobs and calculating their similarities. So the content information plays an important role in the content-based recommendation.

The content-based approach matches candidate profiles with employer profiles and job requirements. Initially based on keyword search, content-based filtering was improved into a statistical inference and semantic engine to figure out relevance rather than match keyword.

Previous studies state that the challenge of matching candidates and jobs is grounded in the interactionist theory of behavior and believe that interactions are important for recommendation as they strongly influence the candidate’s job choice and employer’s hiring decision. This later develops into more sophisticated technique that incorporates content-based relation using structured relevance model and interaction-based relation where applicant actions to items are modeled in a graph.

1. **Collaborative Filtering Approach**

Memory-based approach has 2 types of method. One method is Applicant-Based Collaborative Filtering (UBCF), which calculates applicant-to-applicant correlation to finds similar applicants who have the same taste with the target applicant and recommends items based on what the similar applicants like. Likewise, the other method called Item-Based Collaborative Filtering (IBCF), computes item-to-item correlation to find similarity that may exist among them. The key step in CFR is extracting the similarities among applicants or items

There are again different classifications inside the collaborative filtering. We can have model based or memory based models.

In Memory-based approach the whole set of transactions is stored and is used as the recommender model. These systems employ a notion of distance to find a set of applicants or items, known as neighbours, that tend to agree with the target applicant/item. The preferences of neighbours are then combined to produce a prediction or recommendation for the target applicant/item. The techniques, also known as *nearest-neighbor* or applicant-based collaborative filtering are more popular and widely used in practice.

There are challenges that make this approach may not work well such as data sparsity, scalability and cold start problem.

Another type of collaborative filtering is Model-based. In this approach, collaborative filtering algorithm provides item recommendation by first developing a model of applicant ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of an applicant prediction, given his/her ratings on other items.

The model can be built using machine learning or data mining algorithms like Decision Trees, Neural Network, SVM or rules, which is then used for predictions. Predicting by model gives this approach an advantage in dealing with cold start problem where new applicant or new item introduced in recommender system but this approach tends to limit the range of applicants.

1. **Hybrid Recommendation**

All recommendation approaches mentioned above have their limitations. To overcome the limitation, these approaches have been integrated to obtain better performance. There are seven categories of the hybrid recommender system as follows: weighted, switching, mixed, feature combination, cascade, feature augmentation, and model.

Evaluating the recommender system is a difficult and tricky thing. It is affected by the nature of user that at times users are not only interested in the exact match of their preference but also tend to be interested in discovering something new.

Various technique have been proposed to evaluate recommender system. In general, there are 2 approaches:

1. Relevance Evaluation

Relevance is relation and content similarity between the context of recommended objects and the property of user interest in picking that object. In this evaluation the most relevant items to user’s preference become the basis for evaluation.

1. Predicted Rating evaluation

A Collaborative filtering algorithm is developed to create prediction of user’s rating to items. To measure the performance of the algorithm, a user-rating dataset is split into training/test sets and error is measured on test set predictions after the algorithm has been fed the training ratings [4].

# Method - Hybrid Approach

In this final project the recommendations system is developed using hybrid approach, applying memory based collaborative filtering and content based technique to generate recommended jobs to applicants.

**Memory-Based**

Collaborative filtering technique is applied to find applicants similarity and jobs similarity. These similatries are extracted from the job rating matrix.

The distance between items or applicants is calculated using cosine similarity. From those similarity matrix we then figure out the neighborhood among applicants and jobs.

To improve performance, we reduce the dimensionality of the rating matrix by applying SVD (Singular Value Decomposition) then a number of extracted important features is used in further processing.

**Content-Based**

Relevance between job’s information and applicants profile is discovered using simple keyword matching technique. By doing this, we can improve the accuracy of relevance of jobs to applicant’s preference.

The resulting list from hybrid approach above now become the the base of the recommended jobs, but then we need to exclude all jobs that are already applied by user. We are also applying a proximity analysis to sort the resulting recommended jobs by their distance to applicants.

In this final project we do not generate a predicted rating to jobs that have not been rated by users since in this case we do not have sufficient users that give ratings to more than 8 jobs for the evaluation.

# **Problems with Data**

It is common that performing collaborative filtering will face data sparsity in rating matrix. The dimension of the matrix is very huge since it has a number of users as row and a number of items as column. As most users do not give rating to many items, a sparse rating matrix with many zero values is expected.

High dimensionality can dampen the performance of the algorithm. To overcome this problem we can perform dimensionality reduction technique such as SVD (Singular Value Decomposition) or Matrix Factorization.

Applicants and jobs data happen to mismatch between rating and master data.

Only 924 (out of 6270) jobs in job rating exist in jobs master data.Only 446 (out of 3027) applicants in job rating exist in applicants master data

# Common Problems with Recommenders

Coldstart problem is related to new items or users in rating data. Whenever a new user or new item comes in it lacks information about the rating that it gives or is given to. Therefore, we do not have sufficient information to classify or to perform similarity analysis.

A model based approach is usually applied to classify new items based on trained model. Our approach to overcome this problem is using content-based approach to extract relevance by analyzing the description of an item or profile of a user and classify them accordingly.

When new user or new item comes in, similarity of that user (or item) against all existing users (or items) needs to be calculated. A high number of new items or users combined with huge dimension of rating matrix will take much time to process hence dampen the performance.

Reducing dimensionality will help improve the performance.

# Evaluation of Recommenders

Evaluation of Recommendation engines is based on the underlying evaluation scheme build using the ‘evaluationScheme’ of recommender lab. There are various methods of evaluation scheme

* + - Split
    - Cross Validation
    - Boot strap

We have used both Split and Cross-Validation methods in this project. Cross- Validations are providing better error metrics and we sticked to cross-validation.

The entire rating data is split into three parts, namely

* + - * Train Data - Train data for building model
      * Know Data - Test data with known ratings used for prediction
      * Unknown Data - Test data with unknown ratings for evaluation

For evaluation, the scheme chooses from the test data for each user given items are randomly as "known" items, the remaining items are "unknown." The known items are used to create a prediction and it is evaluated how well the algorithm predicts the unknown items.

Below are the error metrics for split and cross validation methods. Various folds have been tried with cross validation but the error is not improving.

RMSE MSE MAE

SVD1 1.1595961 1.3446630 0.8865419

SVD2 0.9216649 0.8494662 0.6726371

SVD algorithm with different alpha parameter values have been tried with crowd validation methods, but still the error is not improving

# Steps followed to build recommender using SVD

1. Job Rating Matrix

Create job rating matrix from rating data in TrainData.csv file. The job rating matrix has applicant id and job id as the dimensions with the rating data that the applicants give to jobs as the values.

The rows consist of 3027 applicant id’s while the columns consist of 6270 job id’s. And as expected, the matrix is very sparse with only 0.04% of it filled with rating data.

1. Dimensionality Reduction using SVD

Apply SVD (Singular Value Decomposition) to the job rating matrix to extract latent features. Instead of SVD, we can do Matrix Factorization to reduce dimensionality and extract latent features.

1. Predict the top 20 ratings based on above method
2. Use the distance proximity measure for sorting top 20 recommendation to pick the best 10

Distance measure is calculated using the latitude/logitude of Applicant and job.High value has been given to distance if latitude/longitude is not known, so that we can sort jobs on increasing order of distance

Below gives details as how we arrived at distance measure

Latitude Distance = (pi/180)(Latitude 1 - Latitude 2)

Logitude Distance = (pi/180)(Longitude 1 - Longitude 2)

distance = sin(Latitude Distance/2)^2 \* cos(Latitude 1\* pi/180)

\* cos(Latitude 2 \* pi/180)

\* sin(Longitude Distance/2)^2

Multiple above value with 6378.145 to arrive at earth distance between two latitude/logitude values.

# Data Gathering and Understanding the Data

# Data Gathering

Quickhire has collected all the data that are related to jobs and candidates and converted them into comma separated (CSV) files. There are 12 files as follows:

* TrainData.csv

Contains candidate’s rating to the jobs

Format:

*{****Candidate.ID****,* ***Job.ID, Rating****}*

* Main\_Info.csv

Contains basic applicant information about candidates

Format :

*{****Candidate.ID****, City, Zipcode, State.Name, State.Code,* ***Latitude, Longitude****, Estimated.Gender, Estimated.Age, Sign.Up.Date, Authentication.Type, Last.Sign.In.Date, Sign.In.Count, Status,* ***No.Of.Applied.Jobs****, Created.At, Updated.At}*

* Languages.csv

Contains information on language capability of the candidates

Format:

*{Candidate.ID, Language, Spoken.Level, Written.Level, Created.At, Updated.At}*

* Credentials.csv

Contains information on skill and expertise of the candidates

Format:

*{Candidate.ID, Item, Created.At, Updated.At}*

* Leadership.csv

Contains information on leadership quality description of the candidates

Format:

*{Candidate.ID, Item, Created.At, Updated.At}*

* Education.csv

Contains list of education taken by the candidates

Format:

*{****Candidate.ID****, Graduate.Year, School.Name, City,* ***Degree****, Created.At, Updated.At}*

* Experience.csv

Contains history of previous jobs of the candidates

Format:

*{Candidate.ID, Position.Name, Employer.Name, City, State.Name, State.Code, Start.Date, End.Date, Job.Description, Salary, Can.Contact.Employer, Created.At, Updated.At}*

* Interests.csv

Contains list of objects the candidates are interested in

Format:

*{Candidate.ID, Item, Created.At, Updated.At}*

* Positions\_Of\_Interest.csv

Contains expected position from the candidates

Format:

*{****Candidate.ID, Position.Of.Interest****, Created.At, Updated.At}*

* Combined\_Jobs\_Final.csv

Contains job listings with detailed information

Format:

*{Industry,* ***Job.Description****, Requirements, Salary, Listing.Start, Listing.End, Employment.Type,* ***Education.Required****, Created.At, Updated.At}*

* MainJobViews.csv

Contains job page information viewed by candidate

Format:

*{Event.ID, Candidate.ID, Job.ID, Job.URL, Position.Name, Company.Name, Visit.Date.and.Time, Spent.Time,* ***Job.Applied****}*

* Job\_Views.csv

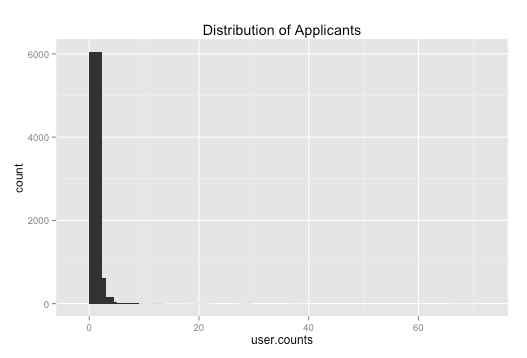
Contains details of job page information

Format:

*{Candidate.ID, Job.ID, Title, Position, Company, City, State.Name, State.Code, Industry, View.Start, View.End, View.Duration, Created.At, Updated.At}*

# Understanding the Data

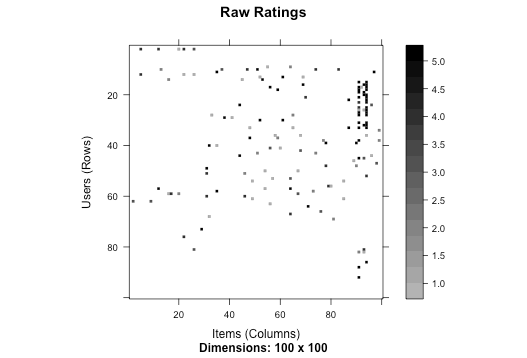
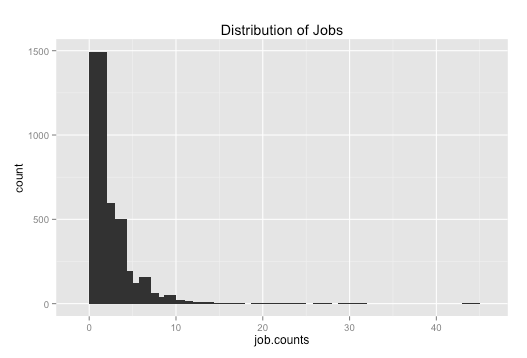
Rating data provided has been converted to “RatingMatrix” format suitable for recommender lab. Rows are applicants and the columns are jobs. We arrived at 3027 X 6270 matrix.

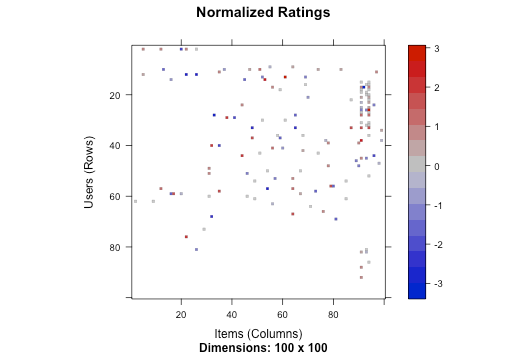
The number of rating in these matrix are very less leading to huge sparsity. Sparsity is the ratio of number of ratings to total possible rating. Only 0.04% of cells in these matrix have value.

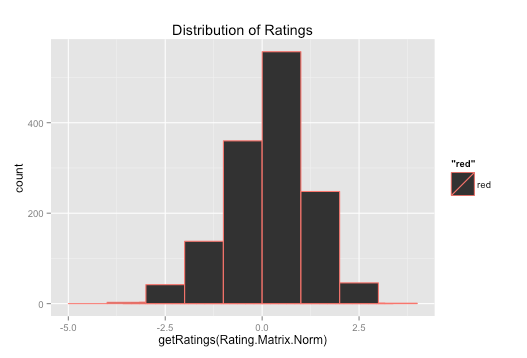
Below plot shows the number of jobs applied by applicants.Almost 6000

jobs applied by less than 3 applicants.

Below plot shows the job distribution, the number of jobs applied by each user.More than 2000 Applicants applied less than 3 jobs

Below image (heat map) shows the rating by applicants. Ignorer to visualise only subset of Applicants and jobs have been considered.

The ratings have been normalised in order to avoid the bias and below image depicts the normalised rating for the same set of users used in above plot.



Plot for frequency of ratings in normalised rating matrix. Ratings have been normalised with mean around 0 and spread from -3 till +3.

# Results

List of top 20 recommendation for first 6 users

> recomm\_matrix.svd.enh[1:6,]

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]

601 "805" "55" "810" "432" "325" "134" "686" "429" "533" "499" "242" "757" "581" "386" "19"

6739 "798" "797" "432" "325" "581" "134" "802" "429" "603" "7" "263" "233" "242" "803" "499"

6808 "624" "581" "787" "283" "612" "802" "499" "263" "803" "233" "654" "512" "6" "788" "736"

6838 "132" "44" "658" "313" "151" "39" "314" "476" "131" "261" "401" "427" "51" "280" "84"

6901 "281" "677" "566" "251" "712" "273" "157" "373" "155" "177" "372" "660" "406" "650" "752"

6915 "90" "155" "152" "251" "712" "407" "566" "395" "156" "650" "158" "76" "273" "770" "416"

[,16] [,17] [,18] [,19] [,20]

601 "564" "227" "507" "551" "802"

6739 "569" "534" "702" "425" "686"

6808 "386" "425" "19" "641" "603"

6838 "197" "256" "32" "615" "216"

6901 "694" "494" "76" "249" "131"

6915 "752" "677" "494" "764" “441"

Now the proximity measure has been applied to filter the recommendation to pick the nest 10 among these. The jobs which have been applied by applicant have also been filtered while picking top 10.

> job.top10[1:6,]

1 2 3 4 5 6 7 8 9 10

601 805 55 810 432 325 134 686 429 533 499

6739 233 798 797 432 325 581 134 802 429 603

6808 233 624 581 787 283 612 802 499 263 803

6838 131 132 44 658 313 151 39 314 476 261

6901 251 131 281 677 566 712 273 157 373 155

6915 251 90 155 152 712 407 566 395 156 650

# Analysis

The data sparsity has become the major problem in building the better recommender system. This has been reduced to less dimension space using the number of rated jobs and number of applicants rated at least a job. So we have reduced 3027 X 6270 original matrix to 469 X 828 matrix.

We normalised these ratings further to avoid any bias by applicants rating all their jobs has high. We then applied the matrix factorisation (Singular Value Decomposition) on this matrix further to build the model.

The top 20 recommended jobs from above model has been enhanced further with proximity measure. Top 20 jobs have been sorted based on the distance between applicant and the job. Jobs with less distance have been picked as top 10 recommended choices.

These recommendations could have been enhanced with

* + - * Comparison between Education Required for Job and Applicants Education
      * Comparison between position of the Job and Applicants interested positions
      * Clustering of users and then recommend jobs based on the cluster the new user belongs to
      * Blending the algorithms - KNN and SVD
      * Mechanisms for reducing the sparsity

# Appendices

<<Not able to Embed the R Script>>

Sending along with This report as separate .R script file.