International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. 03 Special Issue | 2019

Scalable Mobile Crowd sourcing for Urban Laborers and Time Optimization using Machine Learning

Nihit Natu[1], Ayush Gupta[2], Viraj Mahadik[3]
[1, 2, 3] *Department of Computer Engineering Don Bosco Institute of Technology Mumbai, India*

ABSTRACT

The job opportunities in metro cities are immense but finding people with required skillet becomes difficult. Similarly, the workers migrating to these cities find it difficult to get jobs and travel through the city to reach the job location. Our system helps to ease these difficulties by providing jobs to the workers using recommendation systems and finding cost efficient routes from the workers present location to the job site. The recommender system exploits the workers job history to provide recommendations by using concept of Real Boltzmann Machine along with Matrix Factorization and the location services take into consideration various local transport services like shared autos and taxis apart from trains and buses to suggest a cost-efficient path to ease the workers travel in a new city. The deep learning approach further enhances the recommender system efficiency by approximating rank for each job before recommending them to the worker.

Keywords—recommendation systems, deep learning, location services

1. INTRODUCTION

Myriad of laborers migrate to urban cities daily in search of job opportunities. Unaware of the jobs near them, a hefty amount of time is spent in finding work based on their skillet. After locating a job, traversing through a new city may become a problem due to inadequate information about the transport modes in the city. Since most of these are daily workers, they tend to exhaust most of their wages by taking expensive travel modes. Approaching contactors to get jobs leads to lesser pay since the contactor takes a fare amount of share from the workers wage.

Recommendation systems (RS) play an increasingly important role in modern online services. Existing methods of RS can be categorized into two sub types: 1) content- based model and 2) collaborative filtering (CF) based model. Content based methods [9], use the user's history like ratings given to an item by a user and features extracted from the profile to make recommendations. The Content based filtering makes uses of two effective methods: 1) matrix factorization (MF) and 2) restricted Boltzmann machines (RBM). MF directly learns from the latent vectors of the user and the items from the user-item rating matrix and captures the interaction between the user and the item.

2. LITERATURE REVIEW

2.1 Recommender Systems

Recommender Systems are commonly used in social networking platforms like Facebook, Netflix. The most commonly used algorithms for building the recommender systems are Restricted Boltzmann Machine (RBM) and auto Encoders. Various papers related to social media recommendation helped to get in depth working of their recommender engine. Most of them were based of comparing the profiles of various users and finding similarity between them. Pearson's similarity coefficient was commonly used and forms the fundamental for finding the similarity between two profiles [1]. Content and Collaborative based Filtering were two most significant approaches used. Considering the low efficiency provided by the two approaches, Real Boltzmann Machine proved to be suitable for the system. [2], [3]

2.2 Location Services

There are various algorithms used for calculating shortest path between two nodes. The focus while calculating these paths was on the cost factor. Dijkstra's algorithm was fundamental for calculating the cost between nodes although alternations were needed based on our system requirement. Implementing the new feature in the already existing Google Map API's by adding our data based on the local transport services available in a region was the main task that needed extensive research. [8]

3. OVERVIEW OF OUR MODEL

Basically, our system is split into two sections, 1) recommendation systems for providing appropriate jobs based on the workers profile and 2) location services for providing cost efficient routes for traversing from current location to the job location. Fig. 1 shows the prescribed architecture for the system.

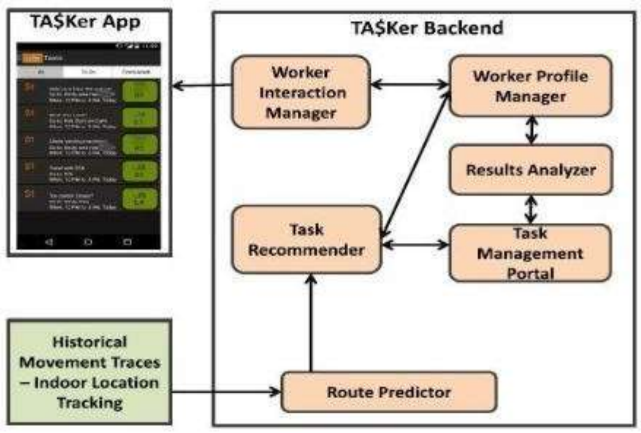

Fig. 1. Architecture of the system.

The application will display the jobs and suggested routes to the user. The Backend consists of recommender systems and location services. The recommendation system algorithms need two sets of inputs. One is the user's dataset which primarily consists of user id, skillets and location and other is the jobs dataset which consists of job id and the rating given by a user to that job. These datasets need to be combined to form a matrix structure with user in its rows and jobs representing the columns. The cells of the matrix will be filled with the ratings given by the user for that job and the jobs not rated with will have a null value.

3.1 Matrix Factorization

A ratings matrix consists of a matrix ***R***, where entries r (i, j) are ratings of user i for item j. These matrices are large, with millions of users and millions of different items. They are also sparse, meaning that each user has typically rated, viewed, or purchased only a small number of items relative to the entire set. The matrix factorization method assumes that there is a set of attributes common to all items, with items differing in the degree to which they express these attributes.

$$R \approx \hat{R} = wu^T \qquad (1)$$

Where R represents the actual rating and $\hat{R}$ is predicted ratings. W and U are used to represent jobs and users respectively.

Calculating the loss function by comparison of the actual and predicted ratings, the efficiency of the system can be found. Minimizing the loss function is main objective of the system.

$$J = \sum_{i,j \in \omega} \left(r_{i_j} - w_i^T u_j\right)^2 \qquad (2)$$

Where J is the loss function, r_{ij} is rating for a job given by A worker in the rating matrix, w_i and u_j represent the job and user index. For minimizing the loss, gradient is calculated, equated with 0 and then solved for individual parameters U and W. Gradient is calculated by applying partial derivative of each parameter u and w on the loss function J and then equated to 0.

For w the partial derivative is given by:

$$\frac{\partial J}{\partial w_i} = 2 \sum_{j \in \varphi_i} \left(r_{i_j} - w_i^T u_j\right)\left(-u_j\right) = 0 \qquad (3)$$

The equation can be altered to represent it with respect to w and since it uses dot products u can be separated since it is a constant for this equation. The equation with respect to w can be given by:

$$w_i = \left(\sum_{j \in \varphi_i} u_j u_j^T \right)^{-1} \sum_{j \epsilon \varphi_i} r_{i_j} u_j \qquad (4)$$

Similarly, a partial derivative with respect to u is calculated and the final equation in terms of u is obtained.

$$\frac{\partial J}{\partial u_j} = 2 \sum_{i \in \varphi_i} \left(r_{i_j} - w_i^T u_j \right) (-w_i) = 0 \qquad (5)$$

$$u_j = \left(\sum_{i \in \varphi_j} w_i w_i^T \right)^{-1} \sum_{i \epsilon \varphi_j} r_{i_j} w_i \qquad (6)$$

3.2 Restricted Boltzmann Machine (RBM)

The Restricted Boltzmann Machine is a generative stochastic artificial neural network that can learn probability distribution over a set of inputs. Fig. 2 shows the basic representation of RBM.

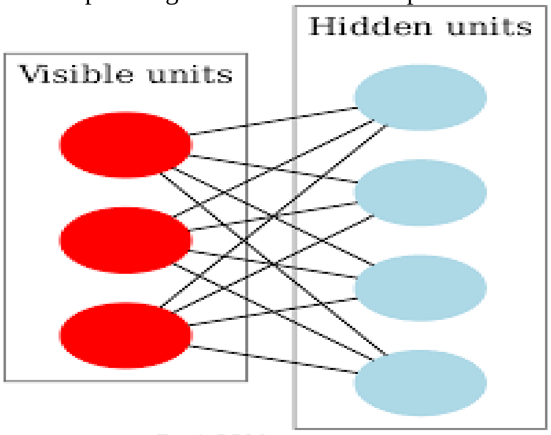

Fig. 2. RBM representation

It consists of visible and hidden nodes and all the nodes are connected with each other. RBM tries to find the hidden nodes representing the features apart from the visible features. The hidden nodes can be calculated from the visible nodes and vice versa. Bernoulli principle is used to calculate hidden nodes values and it can only take values between 0 and 1.

The hidden node can be calculated by:

$$p(h = 1|v) = \sigma(W^T v + b) \qquad (7)$$

The visible can be recalculated after obtaining the hidden node and it is given by:

$$p(v = 1|h) = \sigma\big(W \quad h + c\big) \qquad (8)$$

Where h is hidden node, v is visible node, W is weight matrix, b and c are biases and σ is learning constant. The Maximum Likelihood Function (MLF) is used for the training the RBM. MLF uses the tradition technique of Gaussian distribution which is used for maximizing a given function. MLF is used to maximize p (v) which is given by:

$$W, b, c = argmax_{W,b,c} \log p\,(v; W, b, c) \qquad (9)$$

To maximize p (v) first we need to get the equation for p (v) in terms of terms mentioned above. The equation for p (v) is given by:

$$p(v) = \sum_{h} \frac{1}{Z} \exp(-\{v^T Wh + b^T v + c^T h\}) \quad (10)$$

To find the gradient descent of v, we introduce the concept of Free Energy which is given by f (v). The free energy is beneficial to avoid the RBM to enter in an intractable mode and the calculations are limited to a finite number. Since p (v) is intractable, f (v) helps the system to remain tractable. The free energy function is given by:

$$F(v) = -\log \sum_{h} e^{-E(v,h)} \qquad (11)$$

The gradient descent is calculated for p (v) and the equation obtained is mainly divided into two phases: 1) positive phase and 2) negative phase. The positive term indicates the probability of seeing a visible node more likely compared to the second phase which is the negative phase. The gradient equation is given by:

$$\frac{\partial logp(v)}{\partial \theta} = \frac{\partial F(v)}{\partial \theta} - \sum_{v'} p(v') \frac{\partial F(v')}{\partial \theta} \qquad (12)$$

The derivative of the term v given by v' indicated the lower probability of seeing v. The Markov Chain Monte Carlo (MCMC) helps to limit the equation and it remains tractable since it only has to find 2 values. The first iteration finds the hidden node h from the visible node v and the second Iteration finds the node v' from h and the calculations stop. Gibbs Sampling is used along with MCMC method to implement the finite calculation for the RBM.

3.3 Deep Learning Model

The Deep Learning model helps to enhance the efficiency of the recommender systems. It functions in 2 layers. The representation of Deep Learning Model is shown in Fig. 3

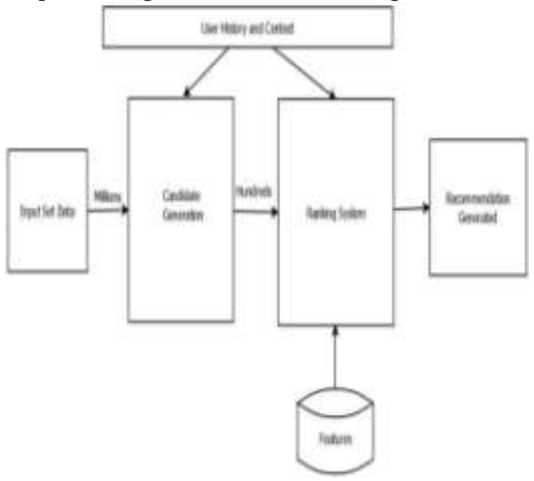

Fig. 3. The Deep Learning Model

It consists of 2 layers, Candidate Generation and Ranking systems that help to improve the existing recommendations.

The model consists of two layers, candidate generation and ranking system. The Candidate generation layers filter the input data set by short listing the selected jobs that can be chosen by the worker. It considers the workers history to shortlist the candidates and it passes this data to the next layer. The Ranking System gives ranking to the shortlisted jobs by considering the features associated with each job like high pay, less travel etc. It also considers the user's history along with the features associated with those jobs and compare to the current available jobs to improvise the shortlisted candidates and enhance the recommendations. [4]

The primary role of ranking is to use impression data to specialize and calibrate candidate predictions for the user interface. For example, a user may choose a prescribed job with high probability generally but is unlikely to choose on the specific job due to the features associated with the job. During ranking, we have access to many more features describing the input and the user's relationship to the job because only a few hundred jobs are being scored rather than the millions scored in candidate generation [5]. Ranking is also crucial for assembling different candidate sources whose scores are not directly comparable. We use a deep neural network with similar architecture as candidate generation to assign an independent score to each job impression using logistic regression. [6]

3.4 Location Services

The next task is to provide the workers with the different transit modes to go the workplace. The route should be chosen in such a way that it helps the workers to reach the place in shortest time and with maximum compensation. The android application of our system will suggest the users the shortest route with the compensation needed for different transit mode.

For reaching from source to any desired place we heavily depend on the map's technology unlike any classical methods of routing through any peers or through asking pedestrian or by following the street notations etc. This may be a way to reach destination if the distance from source to destination is less, but it becomes a serious drawback if the distance is large and the possible nodes or routes are quite large [7]. This is overcome by using any digital Maps technology which are in current demand right now and people mostly rely on this technology in urban areas but rural areas the use of it is seeing a high growth in terms of its usage.

One such method is by using Satellite images which processes population densities of people and houses and maps it as nodes and edges to represent a path between them, but the problem with such technique is that by mere image analysis it is not always guarantee that a path exists between two places by analyzing such satellite images. However, it becomes quite difficult to analyses over 80 billion images and create such routing solutions. The better approach is by using graph theory where vertices and edges are used to create a path.

3.5 Dijkstra's Algorithm

Other solution includes using maps. The underlying algorithmic approach that is used by any digital maps organization like Google Maps, Apple Maps, Here Maps, OpenStreetMap are Dijkstra's routing algorithm. The fact that it is used by such internet giants even after it 50+ years when it was developed, speak the volume about it and its proficiency in calculating the routes. If not Dijkstra's algorithm, then some variations of it are used. We are going to discuss A* search algorithm which is another extension of Dijkstra's. [8]

Algorithm: Cost-efficient route-finding method
Input: Node_current and node_goal
Output: Cost-efficient path

```
1:   Put node_start in the OPEN list with f(node_start) =h(node_start)
2:   while OPEN list is not empty
3:   Take from the open list the node node_current with the lowest
4:   f(node_current)=g(node_current)+ h(node_current)
5:   if node_current is node_goal we have found the solution
6:   Generate each state node_successor that come after node_current
7:   for each node_successor of node_current
8:   Set successor_current_cost = g(node_current) + w(node_current, node_successor)
9:   if node_successor is in the OPEN list
10:  if g(node_successor) <= successor_current_cost
11:  continue
12:  else if node_successor is in the CLOSED list
13:  if g(node_successor) <= successor_current_cost
14:  continue
15:  Move node_successor from the CLOSED list
16:  else Add node_successor to the OPEN list
```

```
17:     Set h(node_successor) to be the heuristic distance to node_goal
18:     Set g(node_successor) = successor_current_cost
19:     Set the parent of node_successor to node_current
20:     Add node_current to the CLOSED list
21:     if (node_current != node_goal) exit with error (the OPEN list is empty)
```

The Dijkstra's algorithm will help to fetch the cost-efficient path from the current location to the destination of the job. The information regarding the local modes of transport needs to be given along with the existing information regarding the bus and train schedules which is implemented in Google Maps. The API modifications will help to implement the local modes of transport to the workers like carpool.

Let us first consider what concept makes the most sense to apply to the problem of finding the shortest path. A network of roads is best displayed in a top-down view, such as in a satellite image. In Figure 4 , we see what a small Swiss town looks like from space. Let us consider what a Swiss town would look like if we were to turn it into a graph. Every intersection and every location of interest can be regarded as a vertex of the graph and the roads that connect vertices are then the edges.

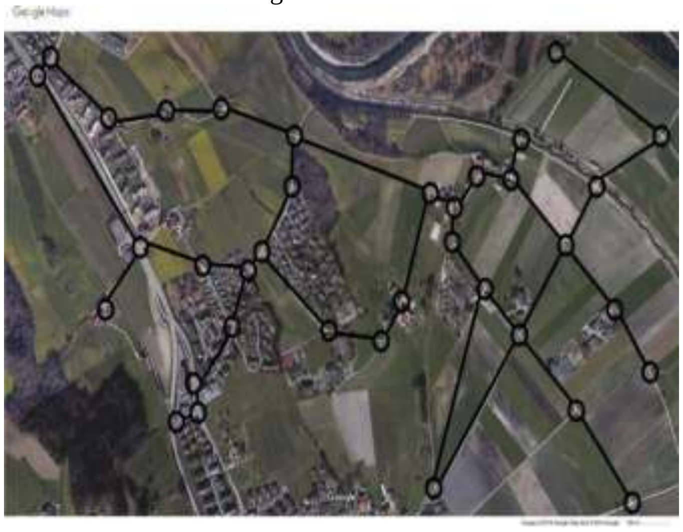

Fig. 4. A small town in Switzerland

Each place a considered a node and the connected map is used to find the shortest path between two nodes. The lines in Figure 4 represent the edges of the graph and the circles are the nodes. Denote the set of edges by E and the set of vertices by V. For each edge $e \in E$ we denote its edge weight by c_e . In the case of a road network, the edge weights represent the time it takes to go from one vertex u to its neighbor v via the edge $\{u,\}$ which represents a road that leads from u to v.In the case of a road network, the edge weights represent the time it takes to go from one vertex u to its neighbor v via the edge $\{u,v\}$ which represents a road that leads from u to v. The task of finding the shortest way from point A to point B can thereby be reduced to finding the shortest path on a weighted graph. There are a lot of different algorithms that can do this, but we only want to discuss the one introduced by Dijkstra.

4. CONCLUSION

The main purpose of this paper is to ease the workers job of finding a job based on their skillset along with a deserved wage for their work. This will help the workers to increase their daily compensation and motivate them to work harder. The travel costs may decrease due to the cheaper routes offered by the system. This system will be more beneficial to a relatively new workers migrated to an urban city in search of jobs. Increase in the number of users will lead to more accurate recommendations from the systems since it can compare various profiles and recommended similar jobs to workers having similar profile but no background history in the system. The job providers will ding it easy to get the workers with best possible skillset. A feedback feature can also be implemented further like the one in Uber, Ola where the driver is given feedback. The workers rating will be used for recommending jobs and users will be provided with highly rated workers. The Deep Learning model suggested by the paper will help the system to evolve and incorporate new features into the system.

5. REFERENCES

[1] Recommender Systems (http://recommendersystems.org/collaborative- filtering/ 2nd October 2018)

[2] Recommender Systems (http://recommender-systems.org/content- based-filtering/ 2nd October 2018)

[3] Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems(https://www.researchgate.net 2nd October 2018)

[4] Deep Learning with Consumer Preferences for Recommender System Tingwei Gao,Xiu Li,Yueting Chai,Youhua Tang,2014

[5] Deep Neural Networks for YouTube Recommendations Paul Covington, Jay Adams, Emre Sargin,2015

[6] A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups Ahmad A. Kardan, Mahnaz Ebrahimi,2017

[7] Google Maps (https://mathsection.com/how-google-maps-calculates- the-shortest-route/?cookie-state-change=1551188613952/ 20th February 2019)

[8] Geo(https://geoawesomeness.com/the-famous-algorithm-that-made- navigation-in-google-maps-a-reality/ 18th February 2019)

[9] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[10] J. Liu, P. Dolan, and E. R. Pedersen, “Personalized news recommendation based on click behavior,” in *Proc. ACM 15th Int. Conf. Intell. User Interfaces*, Hong Kong, 2010, pp. 31–40.