Geo-activity Recommendations by using Improved Feature Combination

Masoud Sattari
Middle East Technical University
Ankara, Turkey
e176326@ceng.metu.edu.tr

Ismail H. Toroslu
Middle East Technical University
Ankara, Turkey
toroslu@ceng.metu.edu.tr

Pinar Senkul
Middle East Technical University
Ankara, Turkey
senkul@ceng.metu.edu.tr

Murat Manguoglu
Middle East Technical University
Ankara, Turkey
manguoglu@ceng.metu.edu.tr

Panagiotis Symeonidis
Aristotle University
Thessaloniki, Greece
symeon@csd.auth.gr

Yannis Manolopoulos
Aristotle University
Thessaloniki, Greece
manolopo@csd.auth.gr

ABSTRACT

In this paper, we propose a new model to integrate additional data, which is obtained from geospatial resources other than original data set in order to improve Location/Activity recommendations. The data set that is used in this work is a GPS trajectory of some users, which is gathered over 2 years. In order to have more accurate predictions and recommendations, we present a model that injects additional information to the main data set and we aim to apply a mathematical method on the merged data. On the merged data set, singular value decomposition technique is applied to extract latent relations. Several tests have been conducted, and the results of our proposed method are compared with a similar work for the same data set.

Author Keywords Geospatial Recommendation, Matrix Factorization, Feature Combination

ACM Classification Keywords H.2.8 [Database Management] Database Applications – data mining

General Terms Algorithms, Experimentation, Performance

INTRODUCTION

With booming technology of smart mobile phones and satellite-assisted positioning systems, new demands for applications in this field are emerging. One of these requirements that most of the users are interested in is activity recommendation based on location (GPS) data, which is specially used to guide tourists and unfamiliar individuals in tourist-attracting cities. Therefore, location-based social networks (LBSN) and location-based recommendations have emerged as interesting research topics involving several dimensions [9, 10, 11]. In this work, we propose a method to improve location-based recommendation by injecting additional data into the original data set. The additional data comes from an external resource into the geospatial activity-location recommendation system.

Our work has actually been inspired from a similar earlier work of [8]. We also use the same data set, including three matrices, namely location-activity matrix, location-feature matrix and activity-activity matrix. The former matrix implies preference ratings of people on a specific activity in a specific location. Its entries actually correspond to the frequency of performing an activity in that location for all users. The second one contains available features of locations and finally, the last one represents relationships among different activities. As expected, the first data set, namely location-activity matrix, is very sparse and we want to determine the values of its unknown entries by utilizing the information in the other two matrices. Generally speaking, merging a data from one resource (Activity-Activity and Location-Feature) with the data from another resource (Location-Activity) is described as *Feature Combination* [1, 7] in the literature. Both [8] and this work apply feature combination to combine these three matrices and construct an integrated matrix. We apply Singular Value Decomposition (SVD) to uncover the latent relations within data. Basic difference and contribution of our work lies in the way integrated matrix is used for prediction and the application of SVD. At the end, we show the effectiveness of our approach by comparing it with the matrix completion approach of [8]. Our experiments show that, our method has higher prediction accuracy than the one in [8].

The rest of the paper has been organized as follows. In Section 2, we discuss related work and consider pros and cons of them. Section 3 explains the data model and we introduce our method in Section 4. Evaluation methods and results of our experiments are available in Section 5. Finally, in Section 6 we have conclusion part of the paper.

RELATED WORK

Nowadays, most of the recommender systems are used in e-commerce to help individuals in decision making [4]. Since social networks like Facebook[1] and Google+[2] have attracted millions of people, several algorithms have been designed to recommend friendship requests and advertisements based on the geographical position of users [2]. Different recommender systems techniques have been proposed in literature and researchers have tried to combine them to build better hybrid models that make use of these techniques [7]. Also, there are applications that track people and get their feedback to build a GPS based recommendation [8].

Some works [9] add another attribute like *user* and model the data with a 3 dimensional tensor in order to have more targeted recommendations using collaborative filtering techniques. Beside this, they use a model-based method that benefits from machine learning techniques, to predict missing values in mentioned tensor. Similarly, the work in [10] presents a mobile recommendation system that, in addition to applying tensor factorization to whole data set, incorporates 2 other algorithms to predict missing values using partial data set.

The aim of our work, as well as the work of [8], can be described as a recommender system that recommends activities for a location. In [8], this is handled by completing the whole activity-location matrix. Thus, when there is a demand for making a recommendation, easily the entry corresponding to that recommendation request can be reached in the completed matrix and the recommendation can be made. However, if that matrix is very large this approach may not be feasible due to two reasons: even with a very effective method, matrix completion may take a lot of time, and there could be a need for a very large storage to store the result. Specially, if the recommendation requests correspond to a small portion of the whole matrix and they sparsely arrive, then responding to those requests on the fly may be a better approach.

In [8], the model that is used to predict unknown values is called *Collective Matrix Factorization*, which was proposed by Singh [6]. Collective Matrix Factorization begins with construction of an objective function. Then, this function is converted to an optimization problem, whereas this task is done iteratively by a numerical method that is known as *Gradient Descent* [5]. Note that this method finds local minima and does not guarantee to find the global minima. Practically, the more local minima are near to global minima, the better the model estimates the unknown values.

Our work aims to tackle the problem of generating recommendations when they are needed. Therefore, dimensionality reduction techniques appear as appropriate approach. In order to be able to deal with huge matrices, we use Singular Value Decomposition (SVD) [3] to generate low rank matrices, while injecting the additional information coming from two other data resources into the main activity-location matrix. Therefore, the main contribution of our work is combining more than one matrix into a single structure. Then, to make a recommendation we use only the main data part of the low rank approximation matrices that have been generated with SVD.

DATA SET CHARACTERISTICS

The data set that is used in this paper is gathered by Microsoft Research Asia. Pre-processed data set is available online and can be downloaded from Microsoft Research Asia website[3]. The data set is collected from a web-based application over 2.5 years so that, each user is equipped with a GPS installed tool (GPS Navigator, smart phone) during visiting Beijing city in China. For each location that is visited, users may insert comments about that place and possible 5 activities that are done in that location (food and drink, shopping, movie and shows, sports and exercise, tourism and amusement) thus, these comments with location info can be used to create a matrix called Location-Activity matrix, whose rows are locations and columns are activities. This matrix consists of 167 locations and 5 activities that each entry of it denotes the frequency of performing an activity for all users in that location.

To make it more informative, location-feature data extracted from *Point of Interest (POI)* database, which is based on city’s yellow pages, is also added to our model. Usually in big cities for any given location area in the city this database gives the type and number of activities like cinemas, restaurants, shopping centers, sport complexes and so on. Gathered data can be modeled as a Location-Feature matrix whose entries are nonnegative integer values that for a given location from available 167 locations shows the frequency of each 13 available features and vice versa [8].

As explained in [8], there is a statistical relation between activities. For example if a user goes to cinema s/he may go to restaurant to eat food too. This is the kind of relationship that is aimed to be captured as Activity Correlation. This information is also available in this data set as a 5-by-5 matrix, which is named as Activity-Activity matrix whose entries are real values in interval [1,-1] and show the correlation between activities represented as the rows and columns.

The aim is mainly to make activity recommendation to the users based on their spatial location. Since, the Location-Activity matrix is very sparse, it makes sense to use the additional information captured in Location-Feature and Activity-Activity matrices in order to make more accurate activity recommendations. That was the main idea behind the work of [8]. Simply, the aim is to make use of Location-

[1] https://www.facebook.com

[2] https://plus.google.com

[3]http://research.microsoft.com/pubs/143146/aaai10.uclaf.data.zip

Feature and Activity-Activity matrices to predict the missing entries of the Location-Activity matrix.

OVERVIEW OF THE PROPOSED METHOD

In this section we describe our proposed method to determine the values of unknown entries of the Location-Activity sparse matrix. In the following sub-sections, we explain the procedure of the merging matrices, construction of the low-rank matrices and Activity-Location recommendation routine. We also describe the whole process through an example thoroughly.

Feature Combination and Low-Rank Matrix Construction

As discussed in Section 3, we have merged main matrix *X* (Location-Activity) with two additional matrices *Y* (Location-Feature) and *Z* (Activity-Activity) to construct an integrated matrix *T*. In order to preserve the structure of a matrix we have injected zero values in the rest of the null entries of *T*.

$$T_{(l+a)\times(f+a)} = \begin{bmatrix} Y_{l\times f} & X_{l\times a} \\ 0_{a\times f} & Z_{a\times a} \end{bmatrix} \tag{1}$$

Then, we simply apply a well-known technique that is *Singular Value Decomposition (SVD)*, to reveal the latent semantic indexing of data. The idea in SVD is to decompose a matrix T into three matrices *U*, S and *V* such that, *U* is left singular matrix, *V* is the right singular matrix and finally *S* contains singular values.

$$T_{r\times s} = U_{r\times r}\ S_{r\times s}\ V^T_{s\times s} \tag{2}$$

In general, SVD is used for dimensionality reduction so that, with selecting the top *k* values of *S* and *k* columns of *U* and V^T and by multiplying them respectively, we can represent matrix *T* with reduced matrix $\mathbb{T}$ which has rank of $k\ (k \le Rank(T))$.

$$\mathbb{T}_k = U_{r\times k}\ \ S_{k\times k}\ \ V^T_{k\times s} \tag{3}$$

For simplicity, in the rest of paper we show the reduced rank components of *T* with U_k, S_k and V^T_k.

Activity Recommendation

Decomposing original matrix *T* reveals an interesting characteristic of U_k and V^T_k. Actually, using U_k for a given activity we can easily find similar locations that this activity is also done and using V^T_k, for a given location we can find similar activities that are done in that location. Moreover, combining these two may even lead to better results. In this step, to predict a frequency rating for a given location *i* and activity *j*, $a_{i,j}$ we search for similar rows of *i* in U_k and also for similar columns of j in V^T_k. In order to have an accurate estimate for frequency rating, instead of selecting one similar neighborhood, we pick the most similar *m* rows in U_k and also the most similar *n* columns in V^T_k. The following equations define these operations.

$$M\_Rating_{sim\ row} = Mean(\textstyle\sum_s Rating(a_{s,j})) \tag{4}$$

$$s = top\ m\ similar\ rows\ in\ U_k$$

$$M\_Rating_{sim\ col} = Mean(\textstyle\sum_s Rating(a_{i,s})) \tag{5}$$

$$s = top\ n\ similar\ columns\ in\ V^T_k$$

After that, the average of $M\_Rating_{sim\ row}$ and $M\_Rating_{sim\ col}$ is used as the predicted rating value for an activity at a location.

Notice that, since the row count of U_k is more than the number of actual locations (due to merge operation) to find similar locations in U_k we trim it so that, its rows corresponds to locations (*l*) only. This is done by selecting the first *l* rows of U_k for the similarity search. Similarly, to find similar activities in V^T_k we trim it so that, its columns corresponds to the activities (*a*) only. It is performed by selecting last *a* columns of V^T_k for the similarity search. Thus, equations (4) and (5) are applied to only these rows and columns.

In order to see influence of similarity metrics, we have applied both Euclidean distance and Cosine similarity to get similarity matrices. On the basis of our experiments, we have observed that, Cosine similarity yields more accurate results than Euclidean distance. Euclidean distance is defined as follows where, $\|\boldsymbol{x}\|$ denotes the Euclidean norm of $\boldsymbol{x}$.

$$sim(\boldsymbol{a},\boldsymbol{b}) = \frac{\boldsymbol{a}^T\boldsymbol{b}}{\|\boldsymbol{a}\|\ \|\boldsymbol{b}\|} \tag{6}$$

Example

In this section we demonstrate a sample example of our method with a simplified data set. As we have already seen, $X_{5\times3}$ is Location-Activity matrix, $Y_{5\times4}$ is Location-Feature matrix and $Z_{3\times3}$ is Activity-Activity correlation matrix.

$$X = \begin{bmatrix} 1 & 3 & 1 \\ 0 & 17 & 17 \\ 0 & 53 & 53 \\ 76 & 4 & 82 \\ 2 & 0 & 0 \end{bmatrix} \tag{7}$$

$$Y = \begin{bmatrix} 0.0037 & 0.0037 & 0.0038 & 0 \\ 0 & 0 & 0 & 0 \\ 0.0082 & 0.0082 & 0.0165 & 0.0131 \\ 0.0100 & 0 & 0.0100 & 0.0318 \\ 0 & 0.0757 & 0 & 0 \end{bmatrix} \tag{8}$$

$$Z = \begin{bmatrix} 1 & 0.0650 & 0.0008 \\ 0.0650 & 1 & 0.0017 \\ 0.0008 & 0.0017 & 1 \end{bmatrix} \tag{9}$$

At the beginning, we combine *X*, *Y* and *Z* matrices based on (1) to construct the integrated matrix *T*. In order to illustrate how our method works and how accurately it determines some missing values, we select 3 nonzero entries from *X* randomly and change their values to 0. The selected entries are x_{11}, x_{22} and x_{33} which are bolded in *T*.

$$T = \begin{bmatrix} 0.0037 & 0.0037 & 0.0038 & 0 & \mathbf{0} & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 17 \\ 0.0082 & 0.0082 & 0.0165 & 0.0131 & 0 & 53 & \mathbf{0} \\ 0.0100 & 0 & 0.0100 & 0.0318 & 76 & 4 & 82 \\ 0 & 0.0757 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0650 & 0.0008 \\ 0 & 0 & 0 & 0 & 0.0650 & 1 & 0.0017 \\ 0 & 0 & 0 & 0 & 0.0008 & 0.0017 & 1 \end{bmatrix} \quad (10)$$

According to (2), SVD method is applied to *T* which yields matrices $U_{8\times8}$, $S_{8\times7}$ and $V^T_{7\times7}$ as follow:

$$U = \begin{bmatrix} \mathbf{-0.025} & \mathbf{-0.010} & -0.999 & 0.035 & 0.001 & -0.016 & 0.005 & 0 \\ \mathbf{-0.279} & \mathbf{-0.124} & 0.008 & -0.031 & 0.952 & -0.008 & 0.008 & 0 \\ \mathbf{-0.870} & \mathbf{-0.385} & 0.026 & 0.010 & -0.305 & 0.003 & -0.003 & 0 \\ \mathbf{-0.405} & \mathbf{0.914} & 0 & -0.026 & -0.001 & 0 & 0.001 & 0 \\ \mathbf{-0.010} & \mathbf{0.024} & 0.035 & 0.999 & 0.033 & 0.001 & 0 & 0 \\ -0.001 & 0.001 & 0 & 0 & 0.006 & -0.302 & -0.953 & -0.001 \\ 0 & 0 & -0.017 & 0 & 0.010 & 0.953 & -0.302 & 0.004 \\ 0 & 0 & 0 & 0 & 0 & -0.004 & 0 & 1 \end{bmatrix} \quad (11)$$

$$S = \begin{bmatrix} \mathbf{79.36} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{75.48} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0076 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.00069 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0000058 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0000017 \end{bmatrix} \quad (12)$$

$$V^T = \begin{bmatrix} -0.00001 & 0.00001 & -0.0002 & -0.001 & \mathbf{-0.36} & \mathbf{-0.63} & \mathbf{0.69} \\ -0.00001 & -0.000002 & -0.00004 & 1 & \mathbf{-0.01} & \mathbf{0.01} & \mathbf{0.004} \\ -0.00002 & 0.000004 & -0.0002 & 0.0005 & -0.73 & -0.27 & -0.63 \\ -0.00003 & 0.00003 & 0.00002 & -0.009 & -0.58 & 0.73 & 0.36 \\ -0.39 & 0.92 & 0.04 & -0.0000002 & 0.00003 & -0.00002 & -0.00002 \\ -0.66 & -0.25 & -0.71 & -0.00004 & 0.0001 & 0.0001 & -0.000008 \\ -0.64 & -0.30 & 0.71 & 0.00002 & -0.0001 & -0.0001 & 0.00001 \end{bmatrix} \quad (13)$$

Since we are interested in specific part of U and V^T, we trim them so that, they meet the same indices of *X* in *T*. In order to reduce the integrated matrix to rank 2, first 2 columns of *U*, *S* and first 2 rows of V^T are selected as shown in (14), (15) and (16) correspondingly.

$$U_k = \begin{bmatrix} -0.025 & -0.01 \\ -0.279 & -0.124 \\ -0.87 & -0.385 \\ -0.405 & 0.914 \\ -0.01 & 0.024 \end{bmatrix} \quad (14)$$

$$S_k = \begin{bmatrix} 79.36 & 0 \\ 0 & 75.48 \end{bmatrix} \quad (15)$$

$$V_k^T = \begin{bmatrix} -0.36 & -0.63 & 0.69 \\ -0.01 & 0.01 & 0.004 \end{bmatrix} \quad (16)$$

To predict the value of x_{ij} in *X* we search for similar rows of *i* in U_k and similar columns of *j* in V_k^T. Remember that to compute similarity matrix, we made use of Cosine similarity between vectors. Related similarity matrices are presented in (17) and (18).

$$Sim(U) = \begin{bmatrix} 1 & 1 & 0.9352 & 0.6974 & 0.9997 \\ 1 & 1 & 0.9356 & 0.6983 & 0.9997 \\ 0.9352 & 0.9356 & 1 & 0.906 & 0.9441 \\ 0.6974 & 0.6983 & 0.906 & 1 & 0.7158 \\ 0.9997 & 0.9997 & 0.9441 & 0.7158 & 1 \end{bmatrix} \quad (17)$$

$$Sim(V^T) = \begin{bmatrix} 1 & 0.998 & 0.511 \\ 0.998 & 1 & 0.461 \\ 0.511 & 0.461 & 1 \end{bmatrix} \quad (18)$$

Using these similarity matrices we compute *Sim_index(U)* and *Sim_index(V^T)*. Each row in *Sim_index (U)* shows the index of most similar rows in descending order from left to right. Similarly, each column in *Sim_index(V^T)* shows the index of most similar columns in descending order from top to down.

$$Sim\_index(U) = \begin{bmatrix} 2 & 5 & 3 & 4 & 1 \\ 1 & 5 & 3 & 4 & 2 \\ 5 & 2 & 1 & 4 & 3 \\ 3 & 5 & 2 & 1 & 4 \\ 2 & 1 & 3 & 4 & 5 \end{bmatrix} \quad (19)$$

$$Sim\_index(V^T) = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad (20)$$

As a final step, we are to predict the value of given x_{ij} from Location-Activity matrix *X*. We find the mean value of top *m* (in this example *m* is chosen to 3) similar nonzero rows of *i* and mean value of top *n* (in this example *n* is chosen to 1) similar nonzero columns of *j* incorporating (19) and (20). The estimated value of x_{ij} is mean value of these 2 terms. Prediction steps for x_{11}, x_{22} and x_{33} are as follow.

Top 3 similar rows of row 1 are 2, 5 and 3 with corresponding values of 0, 2 and 0 in column 1. Since values of rows 2 and 3 are zero we put them aside and select next similar rows which in this case is row 4 only.

$$Mean\ (1, 76) = 38.5 \quad (21)$$

According to first column of (18), column 2 is the most similar column to column 1 with value of 3 in row 1. Thus, the predicted value for x_{11} is mean value of this value and the value that is calculated in (21).

$$Mean\ (38.5, 3) = 19.25 \quad (22)$$

In x_{22}, top 3 nonzero similar rows of row 2 are 1, 3 and 4 with values of 3, 53 and 4 in column 2.

$$Mean\ (3, 53, 4) = 20 \quad (23)$$

Also, column3 is the most similar nonzero column of column 2 with value of 17 in row 2.

$$Mean\ (20, 17) = 18.5 \quad (24)$$

Procedure for x_{33} is same as the previous entries thus, we just show the values.

$$Mean\ (17, 1, 82) = 33.33 \quad (25)$$

$$Mean\ (33.33, 53) = 43.16 \quad (26)$$

In order to show better comparison of predicted values with original values and similar work in [8], we organized them at Table 1.

	Predicted	**Work in [8]**	**Original**
x_{11}	19.25	10.4205	1
x_{22}	18.5	15.3352	17
x_{33}	43.16	7.6185	53

Table 1. Predicted value vs. work in [8] and original value

EXPERIMENTS

In this section we explain the evaluation method and afterwards, present experimental results of our work. Finally we compare the results with similar works.

Evaluation Method

In order to measure the accuracy of our approach, well-known k-fold cross-validation method is used to partition the whole data set into a training data set and a validation data set. To be able to apply this validation technique we have assumed that our data set is accurate. Then, we have set $(1/k)^{th}$ of nonzero entries of the Location-Activity matrix to zero. Afterwards, we have applied our approach on the data set to predict the values of those entries. The k results from the folds then can be averaged (or otherwise combined) to produce a single estimation. In order to compare the results numerically, we have used *Root Mean Squared Error (RMSE)* and *Mean Absolute Error (MAE)* which both measure difference between observed values (original values) and estimated values.

$$MAE(O,E) = \frac{1}{n}\sum_{i=1}^{n}|o_i - e_i| \tag{27}$$

$$RMSE(O,E) = \sqrt{\frac{\sum_{i=1}^{n}(o_i - e_i)^2}{n}} \tag{28}$$

Where, *O* and *E* are observed value and estimated value correspondingly. We have compared our results with a state-of-the-art work that is studied by Zheng et al. in [8].

Experimental Results

In this section we show the experimental results obtained from the proposed model for prediction without applying abstraction method and with abstraction method.

Evaluation without Abstraction

As we discussed in previous subsection, in order to prepare training and validation data, we have used k-fold cross-validation. As it is used usually, we put k = 10 that is, in each fold of execution, we select 10% of nonzero entries in location- activity matrix randomly and put them as validation set, remaining part is the training set and it is used to construct the prediction model.

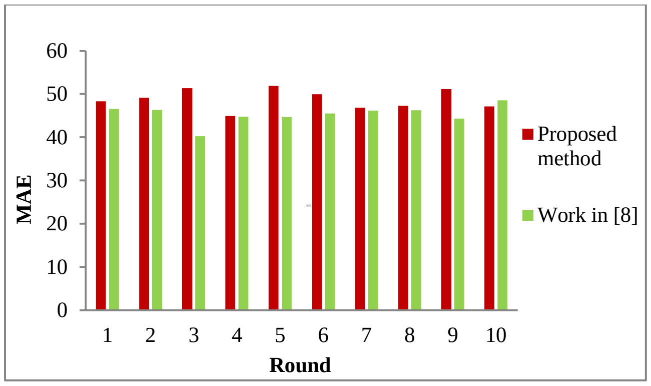

Figure 1. MAE values for proposed work vs. work in [8]

As discussed in Section 3, by using additional data, we implement the proposed method to predict the rating value of all entries in location-activity matrix. Since for validation data we have both observed value and predicted value, we can calculate *RMSE* and *MAE* for this fold. Therefore, in each loop we acquire a value for *RMSE* and *MAE*. At the end of last loop we sum up results for all folds and calculate the mean value of corresponding error terms. Obviously, all steps of 10-fold cross-validation were also applied to recommendation method in [8]. Each column in Figure 1 shows the mean value of *MAE* for 10 folds. In order to have a comprehensive view of results we run the application for 10 times and put the mean value of *MAE* in a new column.

Figure 2 shows the same comparison of RMSE for our proposed method and work in [8]. Like MAE, we run the application for 10 times that each column shows mean value of RMSE for 10 folds.

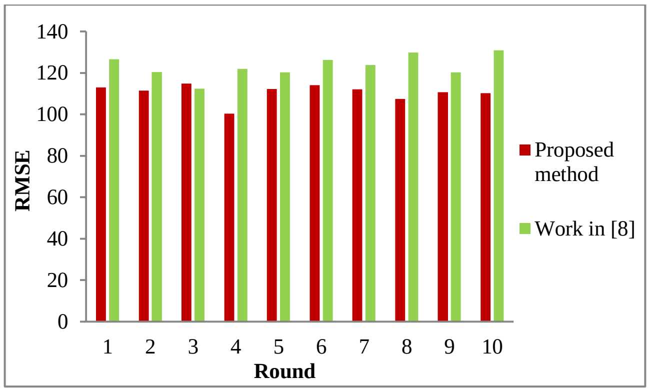

Figure 2. RMSE values for proposed work vs. work in [8]

Evaluation with Abstraction

As mentioned in [8], there are always places that are rated more than other places hence, they tend to have an abnormally greater rating values comparing to other places that are visited frequently but, does not receive much ratings. Beside this, comment adding is an optional case that users are requested to do it and they usually do not provide so many useful comments. In this data set which has 12,765 GPS trajectories 162 users have participated and a total number of 530 comments are collected and based on the previous discussion, most of them are related to some famous places and final Location-Activity matrix becomes very sparse. An explicit drawback of this sparseness is that, the interval between maximum and minimum values is largely extended and moreover, the available frequencies are distributed around specific points within the interval.

Having such a large interval among the values of Location-Activity matrix causes two problems. One is related with the interpretation of these values. Some values are very large, in the range of a few hundreds, and on the other hand, there are also so many values less than 100. It is obvious that the former group corresponds to "strong recommendation". However, without knowing the whole distribution, it is not very easy to determine whether the values in the latter group correspond to "weak recommendation" or "natural". The second problem is related with the error calculation. When the range of values is very large and they are not clustered, their impact in error calculations will be related to their actual values. If there is an activity in a location with frequency 250, and if a system predicts a value such as 300, then this will contribute some error. However, both of these values actually correspond to "strong recommendation". Thus, rather than using these actual values, it would be better if abstract and discrete values are used in a small range. To alleviate the impact of this problem on final evaluations, we propose an abstraction method and calculate error terms to this method respectively.

In this abstraction method, we partition the nonzero values of Location-Activity matrix to 5 clusters using *k*-means clustering algorithm. In this algorithm, 5 random points are selected as initial mean values randomly within the ratings values then, each point (rating value in this case) is assigned to a cluster according to shortest value of Euclidean distance from each mean values. In this step for each cluster, mean value is calculated and assigning points to each cluster is performed iteratively, until no point assigned to a new cluster.

In this abstraction method, instead of working on rating values we have examined the cluster that each value belongs to. Note that, the clusters are ranked in a single dimension. Instead of computing error between original value and predicted value, we find the distance between the clusters that original value belongs to and the cluster that predicted value falls into it. As in previous method, we start this method by applying 10-fold cross-validation but in each fold, instead of keeping the predicted value, we find the cluster for this value. Also, for the similar work in [8], each predicted values fall into one of the clusters. In the last step, we use cluster numbers to calculate the error terms. In our experiments, we have clustered values to 5 clusters. In order to clarify, this procedure is demonstrated for the example in previous section. The nonzero values of matrix X in (7) are clustered to 3 clusters using k-means and corresponding clusters are:

$$C1 = \{1,1,2,3,4,17,17\} \quad C2 = \{53,53\} \quad C3 = \{76,82\} \tag{29}$$

According to (29), validation data $x_{11}=1$, $x_{22}=17$ and $x_{33}=53$ belongs to *C1*, *C1* and *C2* and regarding to Table 1 the predicted values for them are *19.5*, *18.5* and *43.16* which indicates that new clusters for predicted values are *C1*, *C1* and *C2*. As an example, let's assume that for a given validation data x_{ij} the original cluster is *C1*, our predicted value falls into cluster *C3* and the predicted value in work [8] falls into *C2* thus, the MAE error can be calculated consequently as $|1-3| = 2$ for proposed method and $|1-2| = 1$ for similar work in [8].

Figure 3 shows the results of MAE when we apply the abstraction technique. Similar to the previous method each column shows mean value of MAE for 10 folds. We run application for 10 times to show a comprehensive view of results.

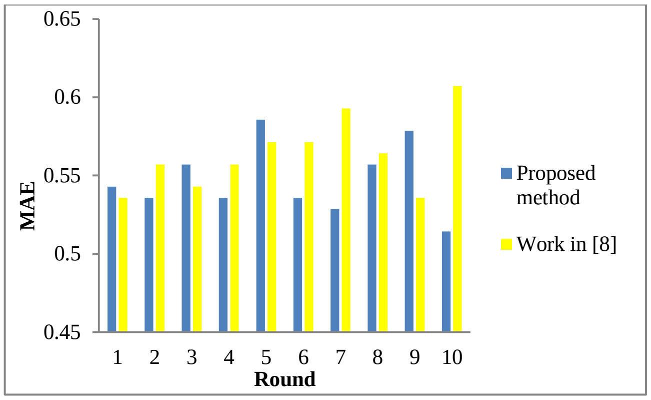

Figure 3. MAE values for proposed work vs. work in [8] with applying abstraction

Finally, Figure 4 shows the mean value of *RMSE* for 10 folds in one column. For the same reason the application is run for 10 times and each column shows mean value of *RMSE* in each time of execution.

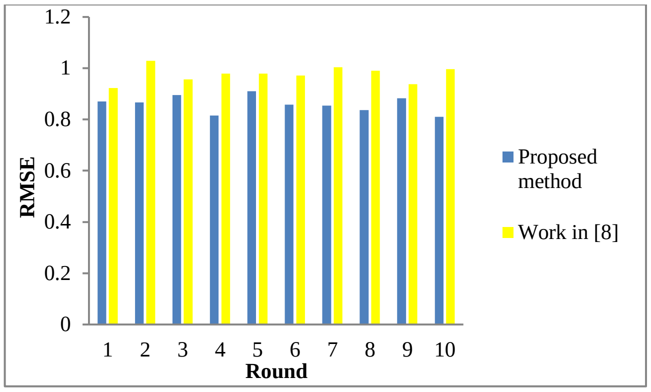

Figure 4. RMSE values for proposed work vs. work in [8] with applying abstraction

Parameter Optimization

Since each data set has its own characteristics, it is not possible to have one set of fixed parameters that works for all. In our work, we have parameterized possible number of top similar locations and top similar activities.

Due to the fact that value of parameters *m* and *n*, which denote the most similar values of rows and columns affect the errors, we have probed different values of these parameters to find the optimal values that reduce the *RMSE*. For each of *m* and *n* we choose values from 1 to 5 thus, we have obtained 25 combinations of them that can be organized in 5 diagrams. We have also implemented it for *RMSE* without abstraction and with abstraction as discussed in previous subsections.

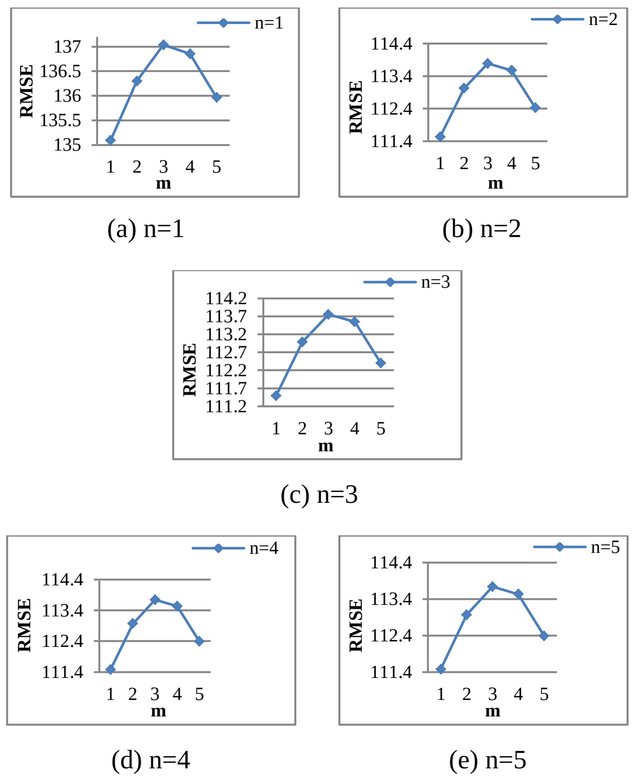

Figure 5. *RMSE* vs. parameter *m* without abstraction

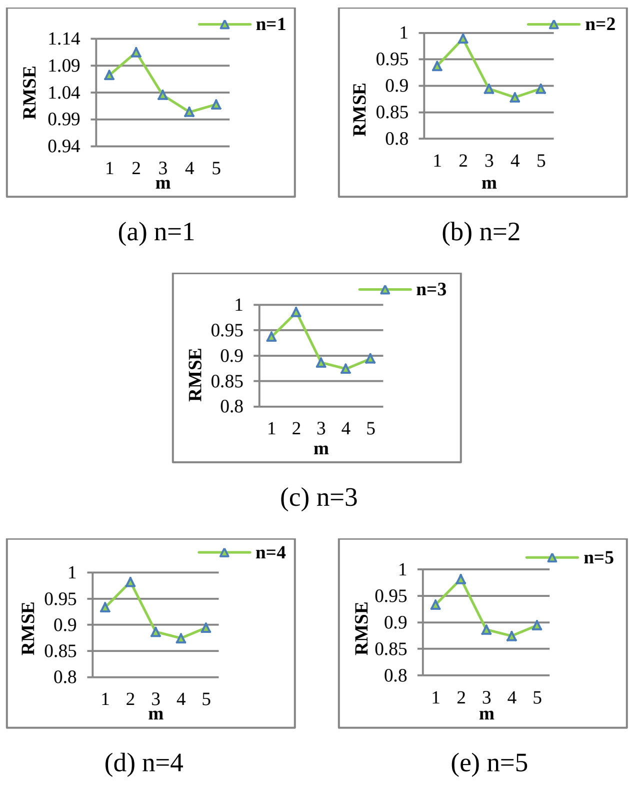

Figure 6. *RMSE* vs. parameter *m* with abstraction

As illustrated in Figure 5, values of *m=1* and *n=3* when results are not abstracted lead to minimum *RMSE*.

Similarly, Figure. 6 shows that *m=4* and *n=4* are the best values when the abstraction is applied, that gives the minimum *RMSE*.

As shown in Figure 5, in this data set for a fixed *n*, RMSE increases up to *m=3*, and then it starts to decrease. However, for even larger values of *m* it does not reach to the level for *m=1*. Therefore, *m=1* seems like the most appropriate choice. Although for different values of *n* slightly different results are obtained, except for 1 they are quite close to each other. Moreover, among them *n=3* gives the smallest *RMSE* values.

On the other hand, when the abstraction is used, for a fixed n, RMSE shows rather fluctuated behavior for increasing *m*. In all the cases we have tested, *m=4* gives the smallest *RMSE* value, and similar to the previous case except for 1, for all other *n* values results are very close to each other, and *n=4* is the smallest among them.

Considering the size of Location-Activity matrix (167 by 5), searching for similar entries more than 5 would not be feasible. It is not possible to interpret why these values produce the best *RMSE* values, but, it is quite clear that it is directly dependent to the data set (the matrix).

CONCLUSION

In this paper, a new approach has been proposed for combining additional information into main sparse data set for making recommendation. This idea has been applied to geo-spatial data set for activity-location recommendation system. Activity correlation data and location feature data has also been added into the system in order to improve the accuracy for predicting the missing values of the sparse location-activity data set. The same problem has already been investigated in [8]. Unlike that work, which aimed to complete the whole matrix, we have used low-rank approximation approach of SVD in order to reply recommendation requests when they arrive. Since the original matrix has been reduced to smaller matrices, its memory requirement and construction times are much less than the method of 7. Furthermore, we use a different method for prediction, which aims to make prediction only cell-wise. This leads to further time efficiency. Indeed, both approaches have their own pros and cons. In large data sets with low recommendation requests our method is more applicable. Moreover, through some experiments, we have also shown that the accuracy values of our approach are better than the one obtained in [8].

ACKNOWLEDGMENTS

This work has been partially funded by the Greek GSRT (project number 10TUR/4-3-3) and the Turkish TUBITAK (project number 109E282) national agencies as part of Greek-Turkey 2011-2012 bilateral scientific cooperation.

REFERENCES

1. Burke, R. D. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, Springer, 2007. 377-408.
2. Huang, Q. and Liu, Y. On geo-social network services. In *Proc. of the 17th IEEE Int. Conf. on Geoinformatics*, Fairfax (VA, USA), 2009.
3. Lax, P. *Linear Algebra and its Applications*, Wiley, 2007.
4. Linden, G., Smith, B. and York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1), 2003.
5. Nocedal, J. and Wright, S. J. *Numerical Optimization.* Springer, 1999.
6. Singh, A. P. and Gordon, G. J. Relational learning via collective matrix factorization. In *KDD '08: Proc. of the 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery & Data Mining*, New York, NY, USA: ACM. 650–658.
7. Spiegel, S., Kunegis, J. and Li, F. Hydra: A Hybrid Recommender System [Cross-Linked Rating and Content Information]. In *CNIKM '09: Proceeding of the 1st ACM international workshop on Complex networks meet information; knowledge management* (2009), 75-80.
8. Zheng, V. W., Zheng, Y., Xie, X. and Yang, Q. Collaborative location and activity recommendations with GPS history data. In *WWW '10: Proc. of the 19th International World Wide Web Conference*. New York, NY, USA: ACM.
9. Zheng, V. W., Cao, B., Zheng, Y., Xie, X. and Yang, Q. Collaborative Filtering Meets Mobile Recommendation: A User-centered Approach. In *AAAI 2010*. ACM, 236-241
10. Zheng, V. W., Zheng, Y., Xie, X. and Yang, Q. Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. In *Artificial Intelligence Journal (AIJ),* 184-185 (2012) 17-37.
11. Zheng, Y. Location-based social networks: Users. In *Computing with Spatial Trajectories*, Zheng, Y. and Zhou, X. Eds. Springer, 2011