ISSN:0975-9646

Implementing Pyramid Structure for Generating Efficient Personnelized Recommendations

Minakshi Pachpatil, Anjana N.Ghule

Computer science and Engineering Department, B.A.M.U.University
Govt.Engg.College,Aurangabad
Maharashtra,India

***Abstract*— In this paper location–aware recommendation system that uses spatial ratings to produce personalized recommendations. System uses Collaborative filtering techniques to generate recommendation based on user location,item location or both user and item location . LARS* uses spatial ratings for spatial items , Spatial ratings for Non-spatial items, Non spatial ratings for spatial items to generate personalized recommendation. For spatial ratings for non spatial items LARS* uses user partitioning technique where spatial ratings are distributed as per user location in the pyramid. Pyramid Maintenance algorithm provided to achieve required scalability or locality. Travel penalty technique is use to find recommended item with minimum distance from querying user. For spatial ratings for spatial items both the techniques are combined with very small modification. LARS* is scalable as number of γ-cells are increased in pyramid and to improve locality α-cells are increased to maintain CF Model. LARS* is efficient as compare to traditional recommendation system because algorithm provided is strong enough to cope challenge of locality and scalability.**

***Keywords*— social ,performance, efficiency, scalability Recommender system, spatial, location.**

I. INTRODUCTION

The goal of recommender system is to generate personalized recommendations for items or products that might interest them. Suggestions for books on Amazon[1],or movies on Netflix are example of recommender system. Recommender systems make use of Collaborative filtering that make use of past community opinions to find similar users or items to generate number of personalized items.Content based filtering that make use of user profile or description of items. Currently working recommendation system make use of (user , item, rating) attributes which is not produce location based recommendations. Spatial recommender system embeds user /item location with ratings e.g. location based social networks(e.g. Forsquare and Facebook Places[4]) allow check-in at spatial destination & rate their visits, Sindbad [3] a location based social network system injects location awarenesss within every aspect of social interaction and functionality in the system.LARS* produces high quality recommendations using Collaborative filtering in an efficient manner. System produces recommendations using three types of spatial ratings in a single framework. (1)Spatial ratings for non spatial items represented using 4-tuple (user, ulocation, item, rating) where ulocation is the user location , e.g. user at home gives rating to the movie/books/restaurant etc.(2)Non-spatial ratings for spatial items having 4 tuples(user, item, ilocation, rating),here item location is specified.e.g. user from unknown location rating a restaurant/hotels. (3) Spatial ratings for spatial items has 5- tuples(user, ulocation, item, ilocation, ratings).e.g. user at his/her location rating a restaurant visted for lunch.

Two techniques that motivate the need for location aware

Preference locality: preference locality suggests that user ratings from one spatial region are different from ratings in another spatial region. Recommendations should be generated by those ratings which are spatially close to the querying user.

Travel Locality: when recommended items are spatial then user has to travel minimum distance when visiting these venues. This property is termed as "travel locality".

II. LARS OVERVIEW

This section provides overview of LARS* query model and Collaborative filtering method.

A. LARS Query Model*

Application has given UserId U, K Numeric limit, Location of the user L, then LARS* generates K-recommended items for querying user. It can support snap shot queries and continuous queries.

B. Item-based Collaborative Filtering.

Main idea of collaborative filtering is to use past opinions of user community to predict which item current user will prefer or interested in. Pure collaborative filtering approach takes user-item matrix as a input and produces following types of output. (1) list of n-recommended items which contains those items that user not purchased before. (2) (numerical) prediction indicating to what degree current user will like the items. In Collaborative filtering assumes a set of n items I={i1,i2,…in} and a set of m users U={u1,u2,…..um}, and each user ui has list of items Iui for which user expressed opinions. Opinions can be numeric rating e.g.(1 to 5 where 1 represents bad choice and 5 represents best choice) or unary rating (e.g.checkboxes) .Active user Ua for whom the collaborative filtering generates the k-recommended items.

It has two forms.

Prediction: Predicted value is specified in same way as opinions are expressed by active user ua. It is the numerical value expressing predicted likeliness of item ij.

Recommendation: A list of items Ir that active user ua will like most it is the items such that Ir ∩ Iua=ϕ.

In item based collaborative filtering prediction is computed using similarity between items. Cosine similarity is used to compute similarity as it produces more accurate result. The similarity between two items a and b is defined as a rating vector $\overline{a}$ and $\overline{b}$ as shown in fig.2. Similarity difference is calculated as follows.

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

The possible similarities are between 0 and 1,where 1 indicates strong similarity. After the similarity between the items are calculated the next stage is to predict the rating using

prediction.For user u and product p prediction is calulated as follows.

$$pred(u, p) = \frac{\sum_{i \in ratedItems(u)} sim(i, p) * r_{u,i}}{\sum_{i \in ratedItems(a)} sim(i, p)}$$

Prediction is the sum of ru,i, a user u's rating for item i ,weighted by sim(i ,p) similarity of product p to candidate item i,then normalized by similarity score between i and p.

User receives as recommendations the top k-item ranked by pred(u ,p).

III. SPATIAL USER RATINGS FOR NON-SPATIAL ITEMS.

LARS* produces recommendations for spatial ratings for non-spatial items using the tuple(user, uloction, item, rating). Main aim is to achieve preference locality i.e. user opinions are spatially unique, To produce recommendations there are 3 requirements. (1)Scalability: System should be scalable as number of users goes on increasing. (2)Locality: while generating the recommendations consider the ratings of those user spatially close to the querying user. (3)Influence: ability to control the size of spatial neighborhood by system users.

User partitioning technique is use to generate recommendations. This technique uses pyramid structure where space is partitioned into different levels of the pyramid as per user location attribute. System then produces recommendations using remaining attributes(user, item, rating). Shape of the pyramid is driven by three goals scalability, influence, locality.

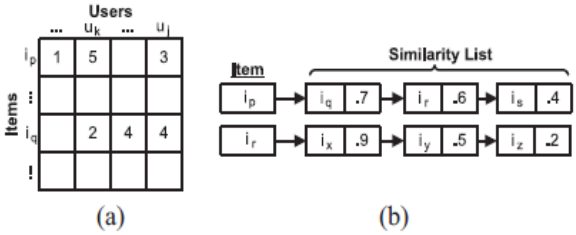


Fig. 1. Item-based CF Model generation.
(a) Rating Matrix. (b)Item based CF Model.

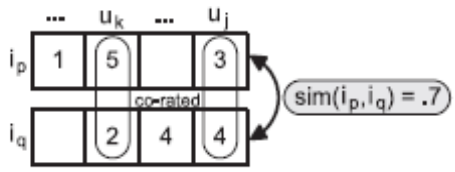


Fig.2. Item-based Similarity calculation.

A. Data Structure.

The pyramid divides the space into h different levels. It make use of partial in-memory pyramid structure[11] as shown in fig.4. For a level h pyramid partitions the space into 4h equal area grid cells. At level 0 (root) representing the total geographic area as a one cell. Level 1 partitioned the space into 4 equi-area cells, and so on. Each cell is represented as by unique cell id. As per the need of recommendation locality and scalability pyramid maintains three types of cells. Empty Cell(γ-cell),Recommendation model cell(α-cell), Statistics Cell(β-cell), Empty Cell(γ-cell).

Recommendation Model Cell(α-cell). Each (α-cell) stores an item based collaborative filtering model. This model is built using spatial ratings located in spatial region of that cell. The α-cellis the root cell of the pyramid and represent traditional item based collaborative filtering model. α-cell maintain items ratings statistics which is in its spatial extents. If Cp is α-cell it contains rating statistics of its 4 cells as shown in fig. 5. For item i1 ratings located in child cells equal to 109,3200,14,54.

Statistics Cell(β-cell). It contains statistics which is within its spatial region. Difference between α-cell and β-cell is that β-cell does not maintains Collaborative filtering model.It is light weight cell as it require less space as compare to α-cell.

Empty Cell(γ-cell). γ-cell is a cell that does not maintain statistics and collaborative filtering model. So it is the most light weight cell as it has no CF model. α- cell is responsible for answering recommendation queries as it contains collaborative filtering model. Pyramid structure that contains only α- cells achives highest locality. Statistics maintained in β- cell determines whether the children of that cell is of α- cells for locality point of view. γ- cells is leaf cells in the pyramid. Based on the tradeoff between locality and scalability cells are upgrade or downgrade. To achieve locality more α- cells are maintained in the pyramid and to achieve scalability more γ-cells are maintained. β –cells comes as intermediary cells which further increase the locality where scalability is not affected.

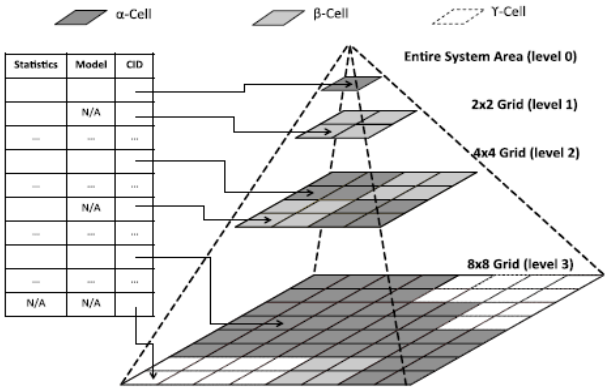


Fig.4 Pyramid Data Structure.

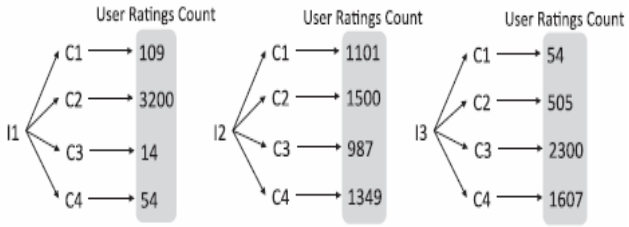


Fig.5 Example of Item Ratings Statistics Table.

B. Data Structure Maintainance.

Root cell in the pyramid represents entire region and cells in lowest level representing more localized regions.Initially pyramid is constructed using all spatial ratings and all the cells in the pyramid is α-cells. Then invoke cell type maintenance step which scans each cells and downgrades cells to (β –cell or γ –cell) if necessary.

As number of users and items added to the system data will goes on increasing. So that size collaborative filtering model as well as number of recommendations produced from each cell also increasing. Cell maintenance is invoked when cell receives N% new ratings, which is computed from the existing ratings.

Features of pyramid maintenance as follows.

(1) Maintenance is performed offline using old pyramid cells.
(2) Maintenance does not reconstruct the whole pyramid at once ,only one cell is rebuilt at a time.
(3)Maintenance is performed only when N% ratings are received by the cell.

C. Maintenance Algorithm.

Algorithm1 provides pseudocode for pyramid maintenance algorithm. It receives input as a Cell C, level h. This algorithm includes three main steps.

Step I: Statistics Maintenance. Parameters in cell C for each item i represents number of user ratings associated with its four children.Item ratings statistics table contained ratings which are in spatial extents of cells.
If N% new ratings received then cell switching decision is made.

Step II: Model Rebuild. As the cell receives new ratings the second step is to rebuild Item based collaborative filtering model. Model is rebuild at cell C only if cell is α-cell. It is necessary to rebuild the model as new ratings enter the system and it should be evovlve in Collaborative filtering model.

Step III: Cell Child Quadrant Maintenance. Pyramid is driven by three goals scalability, influence, locality. Locality and scalability is achieved by cell switching decision. If Cell C child quadrant cells are α-cells then these cells are downgraded to β-cells by calling function CheckDowngradetoScells. If child quadrant cells are β-cells then cells are switched to α-cells by calling function CheckUpgradetoMCells. If child quadrant cells are β-cells then LARS* first considers to swith the cells to α-cells i.e. cell upgradation is done. If cells are not switched to α-cells then downgraded to γ-cells. Cell switching decision is taken completely in quadrants.

1) Recommendation Locality

Running Example. Two level pyramid in which root cell is cp and it is divided into four cells c1,c2,c3,c4 shown in fig.6. Eight users u={u1,u2,…….,u8} and eight items i={i1,i2,….i8}. each user has given ratings to available items. As shown in fig. 6(b) user u2 and u5 belongs to cell c2 both rated the items i2,i5. So the similarity score calculated at c2 is similar to similarity score calculated at parent cell cp. As both the users belongs to same cell. This will not be same if users belong to different cell. LARS* loses the locality if CF model produced at child cells different than the CF model at parent cell. System calculates locality gain/lost as follows.

Locality Loss/Gain. Table2 gives mathematical notions used for calculating locality loss/gain. Items ratings pair set (RPc,i) is a set of pairs of users have rated item I in cell c.

e.g. RPcp,i7 is the item rating pair set for item i7 in cell cp with three elements i.e. RPcp,i7 = { <u3,u6> ,<u3,u7> ,<u6,u7>}. For each item define the Skewed Item Rating Set RSc,I which is total number of user pairs in cell c that rated the item I such that user pairs does not belongs to same cell c. i.e. Skewed item ratings set for i2 in cell cp is null as users u2 and u5 rated item i2 located in same child cell c2. Skewed item ratings set for i4 is RScp,i4={<u2,u7>, <u7,u4>, <u2,u4>}.

Using these parameters calculate the item locality loss LGc,i for each item as follows.

Defination1. Item Locality Loss(LGc,i). it is defined as degree of locality loss of item when four children cell of cell c is downgraded to β-cell , such that 0<=LGc,i<=1.

Defination 2. Locality Loss(LGc). It is defined as total locality loss of cell c by downgrading four children cells to β-cell (0<=LGc<=1). It is the sum of all item locality loss normalized by total number of items in the cell.

cell locality loss determines whether cell children need to downgrade from β-cell to γ-cell ,or upgrade from γ-cell to β-cell, downgraded from α–cells to β-cell.Which are explained as follows.

Summary of Mathematical Notations

Parameter	Description
$RP_{c,i}$	The set of user pairs that co-rated item i in cell c
$RS_{c,i}$	The set of user pairs that co-rated item i in cell c such that each pair of users $\langle u_1, u_2 \rangle \in S_{c,i}$ are not located in the same child cell of c
$LG_{c,i}$	The degree of locality lost for item i from downgrading the four children of cell c to β-Cells, such that $0 \leq LG_{c,i} \leq 1$
LG_c	The amount of locality lost by downgrading cell c four children cells to β-Cells ($0 \leq LG_c \leq 1$)

2) Downgrading α-cells to β-cell.

This operation downgrades the entire cell quadrant at level h from α-cells to β-cell having common parent at level h-1. Downgrading the cells to β-cell improves the scalability with two performance improvement (1)Continuous query processing computation is less as it does not maintain CF model and no need to update recommendation query answer as it crosses boundry of β-cell if it covers large spatial region. (2) low maintenance cost as β- cells does not contain CF model so less CF models are built periodically.

To downgrade the cells to β-cells call the function CheckDowngradeToSCells in Algorithm1. It require two percentage values (1)Locality Loss as described in prevous section.(2) Scalability gain,the amount of scalability gain by downgrading the cells. LARS* Downgrades the cells to β-cell if

*(1-M) * Scalability_gain > M * locality_loss*

M is a real number having range[0,1]. Large value of M is use to achieve locality, Scalability is achieved by small value of M.

Scalability Gain. Scalability gain is calculated by the summation of recommendation model size of cells to be downgraded(sizem) divides this value by sum of sizem and model size of parent cell.

Cost. Using Item Ratings statistics table at cell cp the locality loss is calculated inO(|Icp|) time, where Icp is the total number of items in the cell. Time required to calculate scalability gain is O(1). i.e. total time cost required to downgrade cells to β-cells is O(|Icp|).

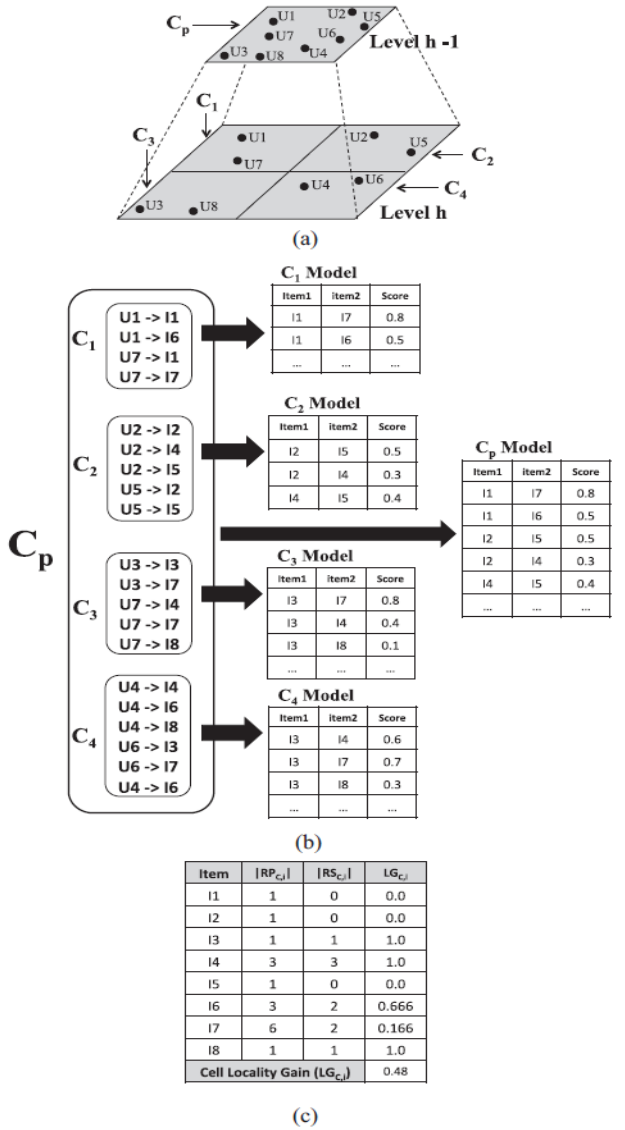
3) Upgrading β-cells to α-cells

Upgrading β-cells to α-cells increase the recommendation locality. It requires to maintain CF model at each children cells which need to be upgrade. It hurts the scalability because more maintenance and storage is required for CF model at each child cell. To determine whether the child cells are need to be upgrade two perecentage values are calculated that is locality gain and scalability loss. Child cells of cell cp are upgrade if following condition holds.

*M * locality gain > (1-M) * Scalability loss.*

It is opposite to downgrading α-cells to β-cell. Locality gain is calculated same way as the locality loss is calculated. Scalability loss is estimation of storage required to maintain CF model at child cells. For item based CF Model the maximum size is n[I] . n is model size. By multiplying n[I] to the number of bytes required to store an item to find upper bound of storage size. Scalability loss is calculated by the sum of These four estimated sizes (sizes) divided by the sum of existing parent cell and sizes.

Time required to calculate scalability loss is O(1) and time required to calculate locality gain is O(|Icp|). Total time cost to upgrade cells to α-cell is O(|Icp|).



Item1	Item2	Score
I1	I7	0.8
I1	I6	0.5
...	...	...

Item1	Item2	Score
I2	I5	0.5
I2	I4	0.3
I4	I5	0.4

Item1	Item2	Score
I1	I7	0.8
I1	I6	0.5
I2	I5	0.5
I2	I4	0.3
I4	I5	0.4
...	...	...

Item1	Item2	Score
I3	I7	0.8
I3	I4	0.4
I3	I8	0.1
...	...	...

Item1	Item2	Score
I3	I4	0.6
I3	I7	0.7
I3	I8	0.3
...	...	...

Item	$\|RP_{c,i}\|$	$\|RS_{c,i}\|$	$LG_{c,i}$
I1	1	0	0.0
I2	1	0	0.0
I3	1	1	1.0
I4	3	3	1.0
I5	1	0	0.0
I6	3	2	0.666
I7	6	2	0.166
I8	1	1	1.0
Cell Locality Gain ($LG_{c,i}$)			0.48

Fig 6.(a) Two Level Pyramid. (c) Locality loss/gain. (b) Recommendation model and Item ratings distribution. The value of RSc,I and RPc,I is derived from item ratings statistics table.Value of LGc,i is used to calculate overall cell locality loss.

4) *Upgrading β-cells to α-cells*

Upgrading β-cells to α-cells increase the recommendation locality. It requires to maintain CF model at each children cells which need to be upgrade. It hurts the scalability because more maintenance and storage is required for CF model at each child cell. To determine whether the child cells are need to be upgrade two perecentage values are calculated that is locality gain and scalability loss. Child cells of cell cp are upgrade if following condition holds.

*M * locality gain > (1-M) * Scalability loss.*

It is opposite to downgrading α-cells to β-cell. Locality gain is calculated same way as the locality loss is calculated. Scalability loss is estimation of storage required to maintain CF model at child cells. For item based CF Model the maximum size is n[I] . n is model size. By multiplying n[I] to the number of bytes required to store an item to find upper bound of storage size. Scalability loss is calculated by the sum of These four estimated sizes (sizes) divided by the sum of existing parent cell and sizes.

Time required to calculate scalability loss is O(1) and time required to calculate locality gain is O(|Icp|). Total time cost to upgrade cells to α-cell is O(|Icp|).

5) Downgrades β-cells to γ-cells and vice versa.

In this β-cells of children quadrant q at level h is downgraded to γ-cells. It has same parent at upper level.It does not required overhead of storing item ratings statistics. It reduce the amount of locality. Decision is taken based on MAX_SLEVEL, maximum number of consecutive pyramid levels. The value is between 0 and total height of the pyramid. A high value of MAX_SLEVEL causes maintaining more β-cell and less γ-cell. As shown in fig. 4 two levels of pyramid are β-cell and third level will automatically set to γ-cell. For each β-cell there is S-Level counter which stores the consecutive number of β-cell levels. If its children cells are β-cell then MAX_SLEVEL is compared with S_LEVEL counter, at a β-cell. The counter counts only when consecutive level contains β-cell if it encounter α-cells counter reset to zero. If S_LEVEL counter is greter than or equal to MAX_SLEVEL then children cells downgraded to γ-cells.Similarly S_LEVEL counter is use to upgrade γ-cells to β-cell.

IV. NON-SPATIAL USER RATINGS FOR SPATIAL ITEMS.

In this recommendations are produced using tuple(user, item , ilocation ,rating). For generating the recommendations for spatial items item based collaborative filtering is used along with travel penalty. It is a technique to limit the user choice venues based on minimum travel distance.

A. Query Processing.

For query processing ,recommendations are generated using item based collaborative filtering to rank the items using travel penalty technique for user u and spatial item i based on RecScore(u,i) computed as follows.

$$RecScore(u,i)=P(u,i) - TravelPenalty(u,i)$$

P(u,i) is the predicted rating of item i for user u. TravelPenalty(u,i) is travel distance between u and i normalize to same value range as rating. When processing the query to compute the travelling distance for all candidate is expensive so it necessary to avoid use of this equation for all computation. To avoid such computation , evaluate items in monotonically increasing order of travel distance.

Algorithm2 gives pseucode for query processing algorithm which takes User u, Location l, Limit k as a input and ge nrates R list of top k-recommended items. Algorithm uses k- nearest algorithm to find the items spatially close to the user, and fills the list R, then R is sorted by recommendation score calculated by equation. Lowest recommendation value is set as RecScore for kth item in list R. Algorithm retrieves item in incremental order of their penalty score using k –nearest neighbor algorithm. For each item i maximum possible recommendation score is calculated by subtracting travel penalty from MAX_RATING. If i is not fit into top k-recommended list with its maximum possible score then algorithm terminates without computing further travel penalty for more items. If early termination case does not arise algorithm continue to compute score for each item and insert into list R.

V. SPATIAL USER RATINGS FOR SPATIAL ITEMS.

Spatial user ratings for Spatial items generates recommendations using tuple(user, ulocation, item, ilocation, ratings). LARS* takes the advantage of using both user partitioning technique and travel penalty for generating recommendations with slight modifications. The data structure and maintenance techniques are same as used in both methods used previously. Only the difference is prediction score p(u,i) used in recommendation score calculation is generated using the collaborative filtering method.

VI. DATASET.

Dataset name MH-Hotels is created which consist of list of hotels in Maharashtra, India. Users list include userid, user name, location.Venues list include hotelid , hotel name, location. Rating list contains userid, venueid and ratings(1 to 5)where 1 is lowest rating and 5 is highest rating. Pyramid is constructed so that it will generate more localized recommendations by maintaining more α-cells using algorithm1.cells are downgrade or upgrade as per the requirement of localization using pyramid maintenanace algorithm.

VII. CONCLUSIONS

.LARS* overcomes the problem faced by traditional recommender system. It uses User partitioning technique and travel penalty to concern with spatial ratings and spatial item ratings. Algorithm provides efficiency for sclability and locality. As CF model is maintained depending upon the number of α-cells in the pyramid it assuares quality of recommendations as well as fast response time as number of user goes on increasing.

REFERENCES

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

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. CSWC*, Chapel Hill, NC, USA, 1994.

[3] The facebook blog. *Facebook Places* [Online]. Available: http://tinyurl.com/3aetfs3

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and pos-sible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[5] *MovieLens* [Online]. Available: http://www.movielens.org/

[6] *Foursquare* [Online]. Available: http://foursquare.com

[7] *New York Times - A Peek into Netflix Queues* [Online]. Available: http://www.nytimes.com/interactive/2010/01/10/nyregion/20100110-netflix-map.html

[8] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel, "LARS: A location-aware recommender system," in *Proc. ICDE*, Washington, DC, USA, 2012.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. Conf. WWW*, Hong Kong, China, 2001.

[10] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. Conf. UAI*, San Francisco, CA, USA, 1998.

[11] W. G. Aref and H. Samet, "Efficient processing of window queries in the pyramid data structure," in *Proc. ACM Symp. PODS*, New York, NY, USA, 1990.

[12] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, no. 1, pp. 1–9, 1974.

[13] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD*, New York, NY, USA, 1984.

[14] K. Mouratidis, S. Bakiras, and D. Papadias, "Continuous monitoring of spatial queries in wireless broadcast environments," *IEEE Trans. Mobile Comput.*, vol. 8, no. 10, pp. 1297–1311, Oct. 2009.

[15] K. Mouratidis and D. Papadias, "Continuous nearest neighbor queries over sliding windows," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 6, pp. 789–803, Jun. 2007.

[16] M. F. Mokbel, X. Xiong, and W. G. Aref, "SINA: Scalable incremental processing of continuous queries in spatiotemporal databases," in *Proc. SIGMOD*, Paris, France, 2004.

[17] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM TOIS*, vol. 22, no. 1, pp. 5–53, 2004.

[18] M. J. Carey and D. Kossmann, "On saying "Enough Already!" in SQL," in *Proc. SIGMOD*, New York, NY, USA, 1997.

[19] S. Chaudhuri and L. Gravano, "Evaluating top-k selection queries," in *Proc. Int. Conf. VLDB*, Edinburgh, U.K., 1999.

[20] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algo-rithms for middleware," in *Proc. ACM Symp. PODS*, New York, NY, USA, 2001.

[21] J. Bao, C.-Y. Chow, M. F. Mokbel, and W.-S. Ku, "Efficient evaluation of k-range nearest neighbor queries in road networks," in *Proc. Int. Conf. MDM*, Kansas City, MO, USA, 2010.

[22] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM TODS*, vol. 24, no. 2, pp. 265–318, 1999.

[23] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis, "Continuous nearest neighbor monitoring in road networks," in *Proc. Int. Conf. VLDB*, Seoul, Korea, 2006.

[24] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM TODS*, vol. 30, no. 2, pp. 529–576, 2005.

[25] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. ICDE*, Heidelberg, Germany, 2001.

[26] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *Proc. Int. Conf. VLDB*, Seoul, Korea, 2006.

[27] N. Bruno, L. Gravano, and A. Marian, "Evaluating top-k queries over web-accessible databases," in *Proc. ICDE*, San Jose, CA, USA, 2002.

[28] P. Venetis, H. Gonzalez, C. S. Jensen, and A. Y. Halevy, "Hyper-local, directions-based ranking of places," *PVLDB*, vol. 4, no. 5, pp. 290–301, 2011.

[29] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-based recommendation system using Bayesian user's preference model in mobile devices," in *Proc. Int. Conf. UIC*, Hong Kong, China, 2007.

[30] *Netflix News and Info - Local Favorites* [Online]. Available: http://tinyurl.com/4qt8ujo

[31] Y. Takeuchi and M. Sugimoto, "An outdoor recommendation system in based on user location history," in *Proc. Int. Conf. UIC*, Berlin, SIGMOD,Germany, 2006.

[32] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with GPS history data," in *Proc. Int. Conf. WWW*, New York, NY, USA, 2010.

[33] M. Ye, P. Yin, and W.-C. Lee, "Location recommendation for location-based social networks," in *Proc. ACM GIS*, New York, NY, USA , 2010.