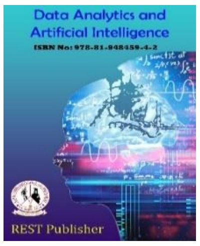
Data Analytics and Artificial Intelligence

Vol: 3(4), 2023

REST Publisher; ISBN: 978-81-948459-4-2

Website: http://restpublisher.com/book-series/daai/


Data Analytics and Artificial Intelligence
ISBN No: 978-81-948459-4-2
REST Publisher

LARS: A Location-Aware Recommender System

*** J Madhavan**

Adhiyamaan College Of Engineering, Hosur, Tamil Nadu, India.

*Corresponding Author Email: madhavanjv1142@gmail.com

Abstract : *This paper proposes LARS, a location-aware recommender system that uses location-based ratings to produce recommendations. Traditional recommender systems do not consider spatial properties of users nor items; LARS, on the other hand, supports a taxonomy of three novel classes of location-based ratings, namely, spatial ratings for non-spatial items, nonspatial ratings for spatial items, and spatial ratings for spatial items. LARS exploits user rating locations through user partitioning, a technique that influences recommendations with ratings spatially close to querying users in a manner that maximizes system scalability while not sacrificing recommendation quality. LARS exploits item locations using travel penalty, a technique that favors recommendation candidates closer in travel distance to querying users in a way that avoids exhaustive access to all spatial items. LARS can apply these techniques separately, or together, depending on the type of location-based rating available. Experimental evidence using large-scale real-world data from both the Foursquare location-based social network and the Movie Lens movie recommendation system reveals that LARS is efficient, scalable, and capable of producing recommendations twice as accurate compared to existing recommendation approaches.*

1. INTRODUCTION

Recommender systems make use of community opinions to help users identify useful items from a considerably large search space (e.g., Amazon inventory [1], Netflix movies [2]). The technique used by many of these systems is collaborative filtering (CF) [3], which analyzes past community opinions to find correlations of similar users and items to suggest k personalized items (e.g., movies) to a querying user u. Community opinions are expressed through explicit ratings represented by the triple (user, rating, item) that represents a user providing a numeric rating for an item. Currently, myriad applications can produce location-based ratings that embed user and/or item locations. For example, location-based social networks (e.g., Foursquare [4] and Facebook Places [5]) allow users to "check-in" at spatial destinations (e.g., restaurants) and rate their visit, thus are capable of associating both user and item locations with ratings. Such ratings motivate an interesting new paradigm of location-aware recommendations, whereby the recommender system exploits the spatial aspect of ratings when producing recommendations. Existing recommendation techniques [6] assume ratings are represented by the (user, rating, item) triple, thus are illequipped to produce location-aware recommendations.

This work is supported in part by the National Science Foundation under Grants IIS-0811998, IIS-0811935, CNS-0708604, IIS-0952977 and by a

In this paper, we propose LARS, a novel location-aware recommender system built specifically to produce high-quality location-based recommendations in an efficient manner. LARS produces recommendations using a taxonomy of three types of location-based ratings within a single framework: (1) Spatial ratings for non-spatial items, represented as a four-tuple (user, ulocation, rating, item), where ulocation represents a user location, for example, a user located at home rating a book; (2) non-spatial ratings for spatial items, represented as a four-tuple (user, rating, item, ilocation), where ilocation represents an item location, for example, a user with unknown location rating a restaurant; (3) spatial ratings for spatial items, represented as a five-tuple (user, ulocation, rating, item, ilocation), for example, a user at his/her office rating a restaurant visited for lunch. Traditional rating triples can be classified as non-spatial ratings for non-spatial items and do not fit this taxonomy. A. Motivation: A Study of Location-Based Ratings: The motivation for our work comes from analysis of two real-world location-based rating datasets: (1) a subset of the well-known MovieLens dataset [7] containing approximately 87K movie ratings associated with user zip codes (i.e., spatial ratings for non-spatial items) and (2) data from the Foursquare [4]. location-based social network containing user visit data for 1M users to 643K venues across the United States (i.e., spatial ratings for spatial items). Appendix B provides further details of both datasets. In our analysis we consistently observed two interesting properties that motivate the need for location-aware recommendation techniques. Preference locality. Preference locality suggests users from a spatial region (e.g., neighborhood) prefer items (e.g., movies, destinations) that are manifestly different than items preferred by users

2. MICROSOFT RESEARCH GIFT

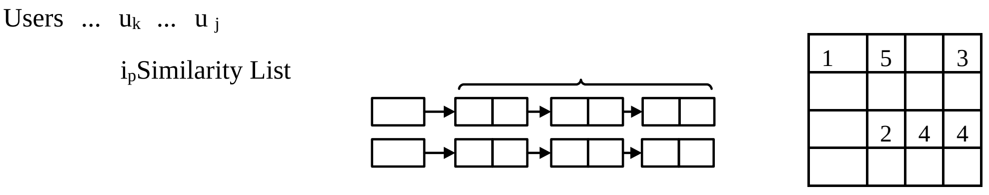
§Work done while at the University of Minnesota

U.S. State	Top Movie Genres	Avg. Rating
Minnesota	Film-Noir	3.8
	War	3.7
	Drama	3.6
	Documentary	3.6
Wisconsin	War	4.0
	Film-Noir	4.0
	Mystery	3.9
	Romance	3.8
Florida	Fantansy	4.3
	Animation	4.1
	War	4.0
	Musical	4.0

Users from:	Visited venues in:	% Visits
Edina, MN	Minneapolis , MN	37 %
	Edina , MN	59 %
	Eden Prarie , MN	5 %
Robbinsdale, MN	Brooklyn Park, MN	32 %
	Robbinsdale, MN	20 %
	Minneapolis, MN	15 %
Falcon Heights, MN	St. Paul, MN	17 %
	Minneapolis, MN	13 %
	Roseville, MN	10 %

(a) Movielens preference locality(b) Foursquare preference locality

FIGURE 1. Preference locality in location-based ratings.



(a) Ratings matrix (b) Item-based CF model

FIGURE 2. Item-based CF model generation.

3. LARS OVERVIEW

This section provides an overview of LARS by discussing the query model and the collaborative filtering method.

A. LARS Query Model

Users (or applications) provide LARS with a user id *U*, numeric limit *K*, and location *L*; LARS then returns *K* recommended items to the user. LARS supports both *snapshot* (i.e., one-time) queries and *continuous* queries, whereby a user subscribes to LARS and receives recommendation updates as her location changes. The technique LARS uses to produce recommendations depends on the type of location-based rating available in the system. Query processing support for each type of location-based rating is discussed in Sections III to V.

B. Item-Based Collaborative Filtering

LARS uses item-based collaborative filtering (abbr. CF) as its primary recommendation technique, chosen due to its popularity and widespread adoption in commercial systems (e.g., Amazon [1]). Collaborative filtering (CF) assumes a set of n users U = $\{u_1,...,u_n\}$ and a set of m items

I = $\{i_1,...,i_m\}$. Each user u_j expresses opinions about a set of items $I_{u_j} \subseteq I$. Opinions can be a numeric rating (e.g., the Netflix scale of one to five stars [2]), or unary (e.g., Facebook "check-ins" [5]). Conceptually, ratings are represented as a matrix with users and items as dimensions, as depicted in Figure 2(a). Given a querying user u, CF produces a set of k recommended items $I_r \subset I$ that u is predicted to like the most.

Phase I: Model Building. This phase computes a similarity score $sim(i_p,i_q)$ for each pair of objects i_p and i_q that have at least one common rating by the same user (i.e., co-rated dimensions). Similarity computation is covered below. Using these scores, a model is built that stores for each item $i \in I$, a list L of similar items ordered by a similarity score *sim*(i_p,i_q), as depicted in Figure 2(b). Building this model is an $O(\frac{R^2}{U})$ process, where *R* and *U* are the number of ratings and users, respectively. It is common to truncate the model by storing, for each list L, only the *n* most similar items with the highest similarity scores [9]. The value of *n* is referred to as the *model size* and is usually much less than |I|.

Phase II: Recommendation Generation. Given a querying user *u*, recommendations are produced by computing *u*'s predicted rating $P_{(u,i)}$ for each item *i* not rated by *u* [9]:

$$P_{(u,i)} = \frac{\sum_{l \in \mathcal{L}} sim(i,l) * r_{u,l}}{\sum_{l \in \mathcal{L}} |sim(i,l)|} \quad (1)$$

Before this computation, we reduce each similarity list L to contain only items *rated* by user *u*. The prediction is the sum

... u_k ... u_j i_p 1
iq

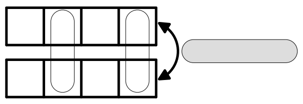


FIGURE 3. Item-based similarity calculation

of $r_{u,l}$, a user *u*'s rating for a related item $l \in$ L weighted by *sim(i,l)*, the similarity of *l* to candidate item *i*, then normalized by the sum of similarity scores between *i* and *l*. The user receives as recommendations the top-*k* items ranked by $P_{(u,i)}$.

Computing Similarity. To compute *sim* (i_p, i_q), we represent each item as a vector in the user-rating space of the rating matrix. For instance, Figure 3 depicts vectors for items i_p and i_q from the matrix in Figure 2(a). Many similarity functions have been proposed (e.g., Pearson Correlation, Cosine); we use the Cosine similarity in *LARS* due to its popularity:

$$sim(i_p, i_q) = \frac{\vec{i_p} \cdot \vec{i_q}}{\|\vec{i_p}\| \|\vec{i_q}\|} \quad (2)$$

This score is calculated using the vectors' co-rated dimensions, e.g., the Cosine similarity between i_p and i_q in Figure 3 is .7 calculated using the circled co-rated dimensions. Cosine distance is useful for numeric ratings (e.g., on a scale [1,5]). For unary ratings, other similarity functions are used (e.g., absolute sum [10]).

While we opt to use item-based CF in this paper, no factors disqualify us from employing other recommendation techniques. For instance, we could easily employ user-based CF [6], that uses correlations between users (instead of items).

4. SPATIAL USER RATINGS FOR NON-SPATIAL ITEMS

This section describes how LARS produces recommendations using spatial ratings for non-spatial items represented by the tuple (user, ulocation, rating, item). The idea is to exploit preference locality, i.e., the observation that user opinions are spatially unique (based on analysis in Section I-A). We identify three requirements for producing recommendations using spatial ratings for non-spatial items: (1) Locality: recommendations should be influenced by those ratings with user locations spatially close to the querying user location (i.e., in a spatial neighborhood); (2) Scalability: the recommendation procedure and data structure should scale up to large number of users; (3) Influence: system users should have the ability to control the size of the spatial neighborhood (e.g., city block, zip code, or county) that influences their recommendations. LARS achieves its requirements by employing a user partitioning technique that maintains an adaptive pyramid structure, where the shape of the adaptive pyramid is driven by the three goals of locality, scalability, and influence. The idea is to adaptively partition the rating tuples (user, ulocation, rating, item) into spatial regions based on the ulocation attribute. Then, LARS produces recommendations using any existing collaborative filtering method (we use item-based CF) over the remaining three attributes (user, rating, item) of only the ratings within the spatial region containing the querying user. We note that ratings can come from users with

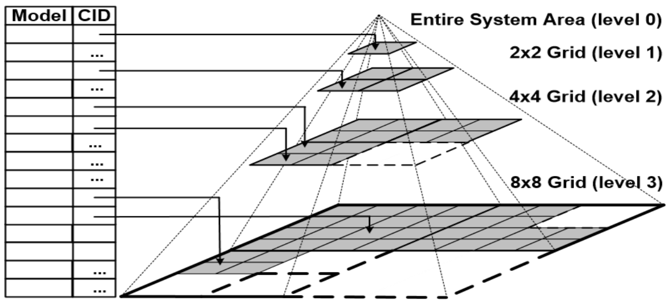
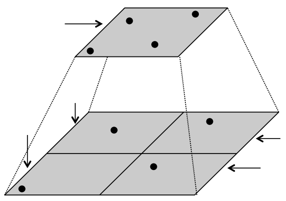


FIGURE 4. Partial pyramid data structure.

varying tastes, and that our method only forces collaborative filtering to produce personalized user recommendations based only on ratings restricted to a specific spatial region. In this section, we describe the pyramid structure in Section III-A, query processing in Section III-B, and finally data structure maintenance in Section III-C.

A. Data Structure

LARS employs a partial pyramid structure [11] (equivalent to a partial quad-tree [12]) as depicted in Figure 4. The pyramid decomposes the space into *H* levels (i.e., pyramid height). For a given level *h*, the space is partitioned into 4^h equal area grid cells. For example, at the pyramid root (level 0), one grid cell represents the entire geographic area, level 1 partitions space into four equi-area cells, and so forth. We represent each cell with the more localized cell $C_u \in q$. Formally, the loss of uniqueness



User	Recommendation		Locality Loss
	C_u	C_p	
U_1	I_1, I_2, I_5, I_6	I_1, I_2, I_5, I_7	25%
U_2	I_1, I_2, I_3, I_4	I_1, I_2, I_3, I_5	25%
U_3	I_3, I_4, I_5, I_6	I_3, I_4, I_5, I_6	0%
U_4	I_3, I_4, I_6, I_8	I_3, I_4, I_5, I_7	50%
Average Locality Loss			25%

FIGURE 5. Merge and split example.

can be computed as the ratio $|R\underline{u}-kR\underline{P}|$, which indicates the number of recommended items that appear in R_u but not in the parent recommendation R_P, normalized to the total number of recommended objects *k*. (3) *Average*. We calculate the average loss of uniqueness over all users in U to produce a single percentage value, termed *locality loss*. Calculating scalability gain. Scalability gain is measured in storage and computation savings. We measure scalability gain by summing the model sizes for each of the merged (i.e., child) cells (abbr. $size_m$), and divide this value by the sum of $size_m$ and the size of the parent cell. We refer to this percentage as the *storage gain*. We also quantify *computation* savings using storage gain as a surrogate measurement, as computation is considered a direct function of the amount of data in the system.

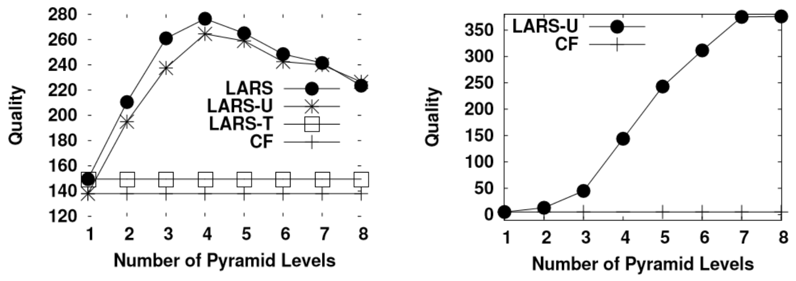
Cost. The cost of *CheckDoMerge* is $|\mathcal{U}|(2(\frac{n|\mathcal{I}|}{4^h})+k)$, where Splitting entails creating a new cell quadrant at pyramid level *h* under a cell at level *h*−1. Splitting improves locality in LARS, as newly split cells represent more granular spatial regions capable of producing recommendations unique to the smaller, more "local", spatial regions. On the other hand, splitting hurts scalability by requiring storage and maintenance of more item-based collaborative filtering models. Splitting also negatively affects continuous query processing, since it creates more granular cells causing user locations to cross cell boundaries more often, triggering recommendation updates.

5. NON-SPATIAL USER RATINGS FOR SPATIAL ITEMS

This section describes how LARS produces recommendations using non-spatial ratings for spatial items represented by the tuple (user, rating, item, ilocation). The idea is to exploit travel locality, i.e., the observation that users limit their choice of spatial venues based on travel distance (based on analysis in Section I-A). Traditional (non-spatial) recommendation techniques may produces recommendations with burdensome travel distances (e.g., hundreds of miles away). LARS produces recommendations within reasonable travel distances by using travel penalty, a technique that penalizes the recommendation rank of items the further in travel distance they are from a querying user. Travel penalty may incur expensive computational overhead by calculating travel distance to each item. Thus, LARS employs an efficient query processing technique capable of early termination to produce the recommendations without calculating the travel distance to all items. Section IV-A describes the query processing framework while Section IV-B describes travel distance computation.

A. Query Processing

Query processing for spatial items using the travel penalty technique employs a single system-wide item-based collaborative filtering model to generate the top-k recommendations by ranking each spatial item i for a querying user u based on RecScore(u,i), computed as:



(a) Foursquare data (b) MovieLens data

FIGURE 6. Quality experiments for varying locality

6. SPATIAL USER RATINGS FOR SPATIAL ITEMS

This section describes how LARS produces recommendations using spatial ratings for spatial items represented by the tuple (user, ulocation, rating, item, ilocation). A salient feature of LARS is that both the user partitioning and travel penalty techniques can be used together with very little change to produce recommendations using spatial user ratings for spatial items. The data structures and maintenance techniques remain exactly the same as discussed in Sections III and IV; only the query processing framework requires a slight modification. Query processing uses Algorithm 2 to produce recommendations. However, the only difference is that the item-based collaborative filtering prediction score P(u,i) used in the recommendation score calculation (Line 16 in Algorithm 2) is generated using the (localized) collaborative filtering model from the partial pyramid cell that contains the querying user, instead of the system-wide collaborative filtering model as was used in Section IV.

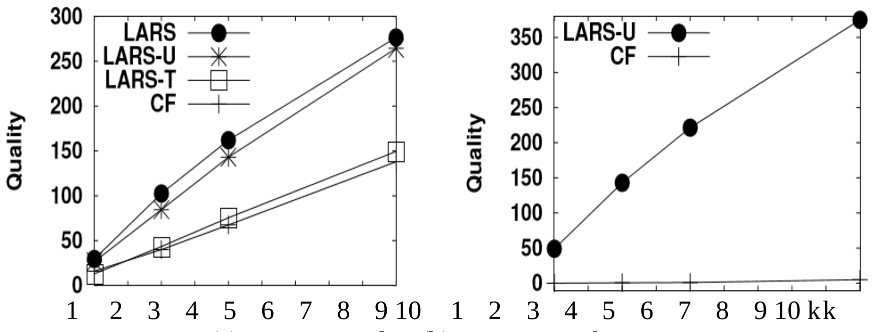
7. EXPERIMENTS

This section provides experimental evaluation of LARS based on an actual system implementation. We compare LARS with the standard item-based collaborative filtering technique along with several variations of LARS. Experiments are based on three data sets: (1) Foursquare: a real data set consisting of spatial user ratings for spatial items derived from Foursquare user histories. (2) MovieLens: a real data set consisting of spatial user ratings for non-spatial items taken from the popular MovieLens recommender system [7]. The Foursquare and MovieLens data are used to test recommendation quality. (3) Synthetic: a synthetically generated data set consisting spatial user ratings for spatial items for venues in the state of Minnesota, USA; we use this data to test scalability and query efficiency. Details of all data sets are found in Appendix B.

Unless mentioned otherwise, the default value of M is 0.3, k is 10, the number of pyramid levels is 8, and the influence level is the lowest pyramid level. The rest of this section evaluates LARS recommendation quality (Section VI-A), trade-offs between storage and locality (Section VI-C), scalability (Section VI-D), and query processing efficiency (Section VI-E).

A. Recommendation Quality for Varying Pyramid Levels

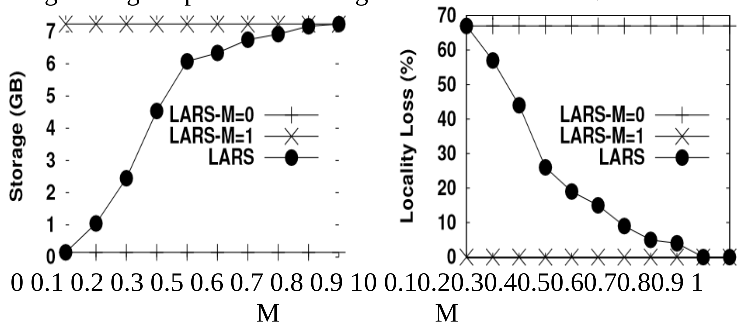
These experiments test the recommendation quality of LARS against the standard (non-spatial) item-based collaborative filtering method (denoted as CF) using both the Fourquare



(a) Foursquare data (b) MovieLens data

FIGURE 7. Quality experiments for varying answer sizes

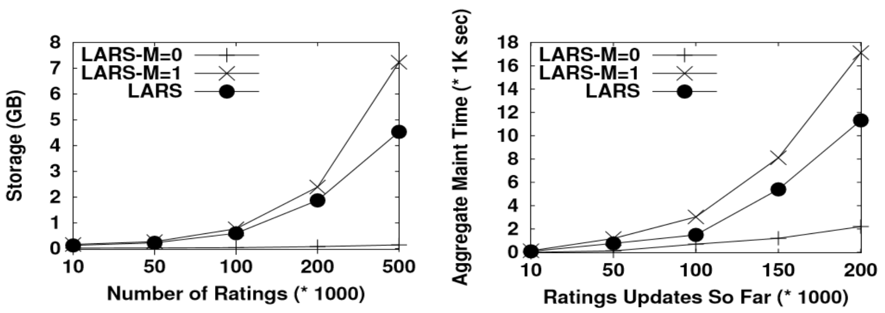
and Movie Lens data. To test the effectiveness of our proposed techniques, we test the quality of Figure 6(b) compares the quality of LARS-U and CF for varying locality using the MovieLens data (LARS and LARST do not apply since movies are not spatial). While CF quality is constant, the quality of LARS-U increases when it produces movie recommendations from more localized pyramid cells. This behavior further verifies that *user partitioning* is beneficial in providing quality recommendations localized to a querying user location, even when items are not spatial. Quality decreases (or levels off for MovieLens) for both LARS-U and/or LARS for lower levels of the adaptive pyramid. This is due to *recommendation starvation*, i.e., not having enough ratings to produce meaningful recommendations.



(a) Storage(b) Locality

FIGURE 8. Effect of M on storage and locality

Recommendation Quality for Varying Values of k These experiments test recommendation quality of LARS, LARS-U, LARS-T, and CF for different values of *k* (i.e., recommendation answer sizes). We perform experiments using both the Foursquare and MovieLens data. Our quality metric is exactly the same as presented previously in Section VI-A. Figure 7(a) depicts the effect of the recommendation list size *k* on the quality of each technique using the Foursquare data set. We report quality numbers using the pyramid height of four (i.e., the level exhibiting the best quality from Section VI-A in Figure 6(a)). For all sizes of *k* from one to ten, LARS and LARS-U consistently exhibit better quality. In fact, LARS is consistently twice as accurate as CF for all *k*. LARST exhibits similar quality to CF for smaller *k* values, but does better for *k* values of three and larger. Figure 7(b) depicts the effect of the recommendation list size *k* on the quality of LARS-U and CF using the MovieLens data (LARS and LARS-T do not apply in this experiment since movies are not spatial). This experiment was run using a pyramid hight of seven (i.e., the level exhibiting the best quality in Figure 6(b)). Again, LARS-U consistently exhibits better quality than CF for sizes of *K* from one to ten. In fact, the quality of CF increases by just a fraction as *k* increases. Meanwhile, the quality of LARS-U increases by a factor of seven as *k* increases from one to ten. C. Storage Vs. Locality Figure 8 depicts the impact of varying M on both the storage and locality in LARS. We plot LARS-M=0 and LARSM=1 as constants to delineate the extreme values of M, i.e., M=0 mirrors traditional collaborative filtering, while M=1 forces LARS to employ a complete pyramid. Our metric for locality is *locality loss* (defined in Section III-C1) when compared to a complete pyramid (i.e., M=1). LARS-M=0 requires the lowest storage overhead, but exhibits the highest locality loss, while LARS-M=1 exhibits no locality loss but requires the most storage. For LARS, increasing M results in increased storage overhead since LARS favors splitting, requiring the maintenance of more pyramid cells each with its own collaborative filtering model. Meanwhile, increasing M results in smaller locality loss as LARS merges less and maintains more localized cells. The most drastic drop in locality loss is between 0 and 0.3, which is why we chose M=0.3 as a default.



(a) Storage(b) Maintenance

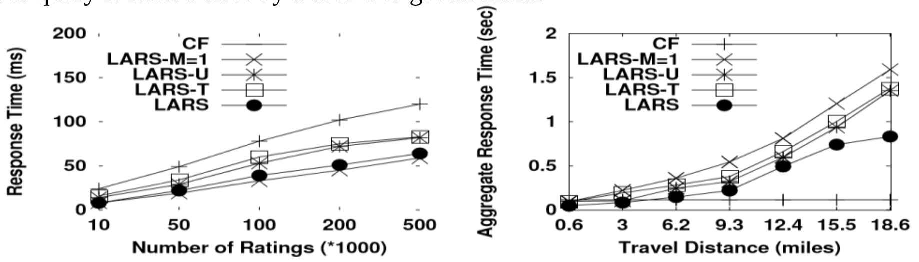
FIGURE 9. Scalability of the adaptive pyramid

Scalability Figure 9 depicts the storage and aggregate maintenance overhead required for an increasing number of ratings. We again plot LARS-M=0 and LARS-M=1 to indicate the extreme cases for LARS. Figure 9(a) depicts the impact of increasing the number of ratings from 10K to 500K on storage overhead. LARS-M=0 requires the lowest amount of storage since it only maintains a single collaborative filtering model. LARSM=1 requires the highest amount of storage since it requires storage of a collaborative filtering model for all cells (in all levels) of a complete pyramid. The storage requirement of LARS is in between the two extremes since it merges cells to save storage. Figure 9(b) depicts the cumulative computational overhead necessary to maintain the adaptive pyramid initially populated with 100K ratings, then updated with 200K ratings (increments of 50K reported). The trend is similar to the storage experiment, where LARS exhibits better performance than LARS-M=1 due to merging. Though LARS-M=0 has the best performance in terms of maintenance and storage overhead, previous experiments show that it has unacceptable drawbacks in quality/locality.

E. Query Processing Performance

Figure 10 depicts snapshot and continuous query processing performance of LARS, LARS-U (LARS with only user partitioning), LARS-T (LARS with only travel penalty), CF (traditional collaborative filtering), and LARS-M=1 (LARS with a complete pyramid).

Snapshot queries. Figure 10(a) gives the effect of various number of ratings (10K to 500K) on the average snapshot query performance averaged over 500 queries posed at random locations. LARS and LARS-M=1 consistently outperform all other techniques; LARS-M=1 is slightly better due to recommendations always being produced from the smallest (i.e., most localized) CF models. The performance gap between LARS and LARS-U (and CF and LARS-T) shows that employing the travel penalty technique with early termination leads to better query response time. Similarly, the performance gap between LARS and LARS-T shows that employing user partitioning technique with its localized (i.e., smaller) collaborative filtering model also benefits query processing. Continuous queries. Figure 10(b) provides the continuous query processing performance of the LARS variants by reporting the aggregate response time of 500 continuous queries. A continuous query is issued once by a user u to get an initial



(a) Snapshot Queries (b) Continuous Queries

FIGURE 10. Query Processing Performance.

answer, then the answer is continuously updated as u moves. We report the aggregate response time when varying the travel distance of u from 1 to 30 miles using a random walk over the spatial area covered by the pyramid. CF has a constant query response time for all travel distances, as it requires no updates

8. RELATED WORK

Location-based services. Current location-based services employ two main methods to provide interesting destinations to users. (1) KNN techniques [19] and variants (e.g., aggregate KNN [21]) simply retrieve the k objects nearest to a user and are completely removed from any notion of user personalization. (2) Preference methods such as skylines [22] (and spatial variants [23]) and location-based top-k methods [24] require users to express explicit preference constraints. Conversely,

LARS is the first location-based service to consider implicit preferences by using location-based ratings to help users discover new and interesting items.
Recent research has proposed the problem of hyper-local place ranking [25]. Given a user location and query string (e.g., “French restaurant”), hyper-local ranking provides a list of top-k points of interest influenced by previously logged directional queries (e.g., map direction searches from point A to point B). While similar in spirit to LARS, hyper-local ranking is fundamentally different from our work as it does not personalize answers to the querying user, i.e., two users issuing the same search term from the same location will receive exactly the same ranked answer set. Traditional recommenders. A wide array of techniques are capable of producing recommendations using non-spatial ratings for non-spatial items represented as the triple (user, rating, item) (see [6] for a comprehensive survey). We refer to these as “traditional” recommendation techniques. The closest these approaches come to considering location is by incorporating contextual attributes into statistical recommendation models (e.g., weather, traffic to a destination) [26]. However, no traditional approach has studied explicit location-based ratings as done in LARS. Some existing commercial applications make cursory use of location when proposing interesting items to ratings. More importantly, LARS is a complete system (not just a recommendation technique) that employs efficiency and scalability techniques (e.g., merging, splitting, early query termination) necessary for deployment in actual large-scale applications.

9. CONCLUSION

LARS, our proposed location-aware recommender system, tackles a problem untouched by traditional recommender systems by dealing with three types of location-based ratings: spatial ratings for non-spatial items, non-spatial ratings for spatial items, and spatial ratings for spatial items. LARS employs user partitioning and travel penalty techniques to support spatial ratings and spatial items, respectively. Both tech-niques can be applied separately or in concert to support the various types of location-based ratings. Experimental analysis using real and synthetic data sets show that LARS is efficient, scalable, and provides better quality recommendations than techniques used in.

REFERENCES

[1]. Levandoski, Justin J., Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. "Lars: A location-aware recommender system." In 2012 IEEE 28th international conference on data engineering, pp. 450-461. IEEE, 2012.
[2]. Sarwat, Mohamed, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. "Lars*: An efficient and scalable location-aware recommender system." IEEE Transactions on Knowledge & Data Engineering 26, no. 06 (2014): 1384-1399.
[3]. Kurinjimalar Ramu, M. Ramachandran, Vimala Saravanan, Manjula Selvam, and Sowmiya Soundharaj. "Big Data Analytics for Mobility Prediction and Its Classification‖." Data Analytics and Artificial Intelligence 2, no. 2 (2022): 74-81.
[4]. Yang, Wan-Shiou, Hung-Chi Cheng, and Jia-Ben Dia. "A location-aware recommender system for mobile shopping environments." Expert Systems with Applications 34, no. 1 (2008): 437-445.
[5]. Rodriguez-Hernandez, M. D. C., Sergio Ilarri, R. Hermoso, and R. Trillo-Lado. "Location-aware recommendation systems: Where we are and where we recommend to go." In CEUR workshop proc., no. ART-2015-92063. 2015.
[6]. Venkateswaran, C., M. Ramachandran, Kurinjimalar Ramu, Vidhya Prasanth, and G. Mathivanan. "Application of simulated annealing in various field." Materials and its Characterization 1, no. 1 (2022): 01-08.
[7]. Ma, Xindi, Hui Li, Jianfeng Ma, Qi Jiang, Sheng Gao, Ning Xi, and Di Lu. "APPLET: A privacy-preserving framework for location-aware recommender system." Science China Information Sciences 60, no. 9 (2017): 1-16.
[8]. Saravanan, Vimala, M. Ramachandran, and Chandrasekar Raja. "A Study on Aircraft Structure and Application of Static Force." REST Journal on Advances in Mechanical Engineering 1, no. 1 (2022): 1-6.
[9]. Ho, Shen-Shyang, Mike Lieberman, Pu Wang, and Hanan Samet. "Mining future spatiotemporal events and their sentiment from online news articles for location-aware recommendation system." In Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, pp. 25-32. 2012.
[10]. Li, Xinyu, Zhongchun Mi, Zhenmei Zhang, and Jiani Wu. "A location-aware recommender system for tourism mobile commerce." In The 2nd international conference on information science and engineering, pp. 1709-1711. IEEE, 2010.
[11]. Chidambaram, P. K., Dr Amol Lokhande, Dr M. Ramachandran, Vimala Saravanan, and Vidhya Prasanth. "A Review on Biodiesel Properties and Fatty acid composites." REST Journal on Emerging trends in Modelling andManufacturing 7, no. 3 (2021): 87-93.