LARS
: A Location-Aware Recommender System

데이터사이언스대학원 석사과정 최영은

CONTENTS

- Abstract
- Introduction
- Spatial User Ratings for Non-Spatial Items
- Non-Spatial User Ratings for Spatial Items
- Experiments
- Conclusions

Abstract

- A taxonomy of three novel classes of location-based ratings

  1) Spatial ratings for non-spatial items (MovieLens)

  2) Non-spatial ratings for spatial items

  3) Spatial ratings for spatial items (Foursquare)

- User partitioning

  - exploiting user rating locations

- Travel penalty

  - exploiting item locations

Introduction

- A taxonomy of three novel classes of location-based ratings

  1) Spatial ratings for non-spatial items (user, ulocation, rating, item)

  2) Non-spatial ratings for spatial items (user, rating, item, ilocation)

  3) Spatial ratings for spatial items (user, ulocation, rating, item, ilocation)

- Motivation: A Study of Location-Based Ratings

  - preference locality

  : influences recommendation using the unique preferences found within the spatial region containing the user

  - travel locality

  : recommendation loses efficacy the further a querying user must travel to visit the destination.

Introduction

- Contributions of LARS

  - A novel location-aware recommender system capable of using three classes of location-based ratings

  (a) a user partitioning technique

  : exploiting user locations in a way that maximizes system scalability while not sacrificing recommendation locality

  (b) a travel penalty technique

  : exploiting item locations and avoiding exhaustively processing all spatial recommendation candidates

  - Experimental evidence that LARS scales to large-scale recommendation scenarios and provides better quality recommendations than traditional approaches

LARS Overview

- LARS Query Model

  - input : U(user id), K(numeric limit), L(location) ☞ output : K recommended items

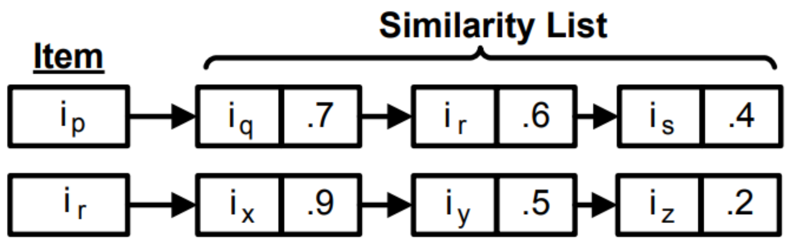
  - snapshot queries & continuous queries

- Item-Based Collaborative Filtering

  - Phase I: Model Building

  : a model is built that stores for each item i ∈ I, a list L of similar items ordered by a similarity score sim(i_p, i_q)

  - Phase II: Recommendation Generation



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

Spatial User Ratings for Non-Spatial Items

- Three requirements for producing recommendations

  (1) Locality

  - a spatial neighborhood

  : ratings with user locations spatially close to the querying user location

  (2) Scalability

  - the recommendation procedure and data structure should scale up to large number of users

  (3) Influence

  : controlling the size of the spatial neighborhood (city block, zip code, or county)

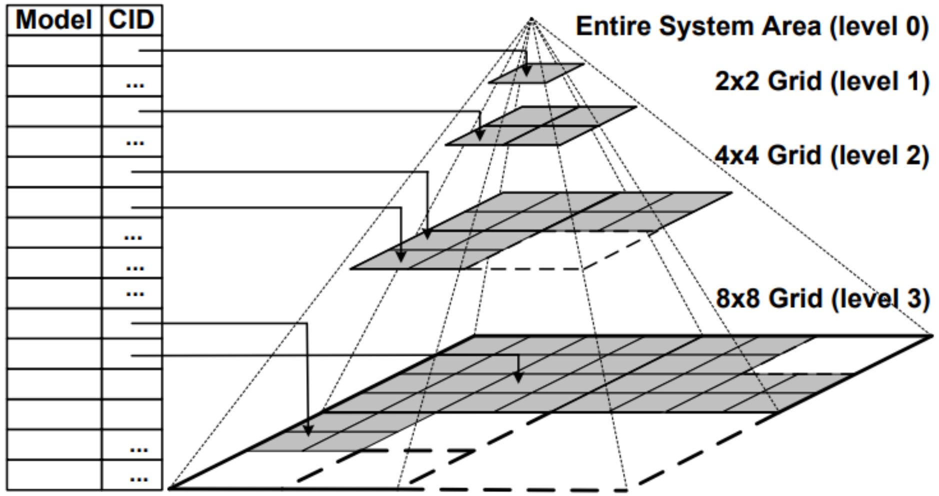
Spatial User Ratings for Non-Spatial Items

- Data Structure

  - For a given level h, the space is partitioned into 4^h equal area grid cells.

  - In each cell, we store an item-based collaborative filtering model built using only the spatial ratings with user locations contained in the cell's spatial region.

  - the root cell (level 0) = a "traditional" (i.e., non-spatial) item-based CF model



Spatial User Ratings for Non-Spatial Items

- Query Processing

(1) Find the lowest maintained cell C in the adaptive pyramid that contains L

(2) The top-k recommended items are generated using the model stored at C.

- Continuous queries

: User crossing a cell boundary → Recommendation result updated

: A cell at level h is not maintained → Go higher and find the nearest maintained ancestor cell

- Influence level

default : Starting from the lowest maintained grid cell

→ Starting from the grid cell containing the querying user location at level l

Spatial User Ratings for Non-Spatial Items

- Data Structure Maintenance

  - all location-based ratings currently in the system are used to build a complete pyramid of height H

  → merging step : quadrants (i.e., four cells with a common parent) at level h into their parent at level h − 1

  → maintenance on a cell-by-cell basis once it receives N% new ratings

  : tradeoffs in scalability and locality

  : checking (1) cell C has a child quadrant q maintained at level h + 1

  : checking (2) none of the four cells in q have maintained children of their own

  ⇒ Yes! quadrant q = a candidate to merge into its parent cell C

  ⇒ No! cell C = a candidate to be splited into four child cells at level h+ 1

Spatial User Ratings for Non-Spatial Items

- Cell Merging

  - discarding an entire quadrant of cells at level h with a common parent at level h－1

  - scalability ↑, locality ↓

  - calculation locality_loss, scalability_gain

  - (1 － M) ∗ scalability gain > M ∗ locality loss

  - M = 0 : a traditional CF ↔ M = 1 : maintaining all cells at all levels (no merging)

- Calculating Locality Loss

  (1) Sample : from users who have at least one rating within C_P

  (2) Compare : R_p(from the merged cell C_p) vs. R_u(from the localized cell $C_u \in q$)

  $$\frac{|R_u - R_P|}{k}$$

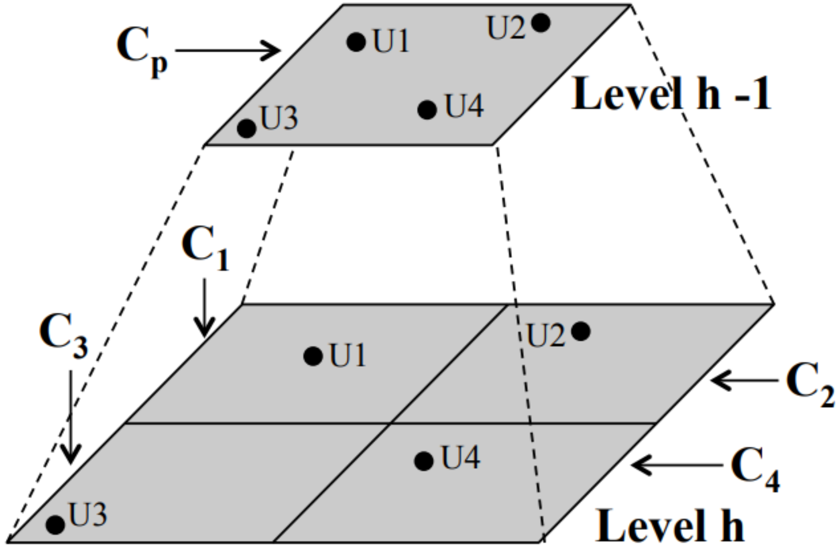
  (3) Average : average loss of uniqueness over all users in U

Spatial User Ratings for Non-Spatial Items

- Calculating scalability gain

  (1) $size_m$: summing the model sizes for each of the child cells

  (2) $size_m$ / ($size_m$ + the size of the parent cell)



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%

Spatial User Ratings for Non-Spatial Items

- Cell Splitting

  - creating a new cell quadrant at pyramid level h under a cell at level h－1

  - scalability ↓ , locality ↑

  - calculation locality_gain, scalability_loss

  - M ∗ locality gain > (1 － M) ∗ scalability loss

  - Speculative splitting

  : building each model using a random sample of only 50% of the ratings from the spatial region of each potentially split cell

- Calculating locality gain

  : if any of the speculatively split cells do not contain ratings for enough unique items

  → immediately set the locality gain to 0 (preventing recommendation starvation)

Spatial User Ratings for Non-Spatial Items

- Calculating scalability loss

  - estimating the storage necessary to maintain the newly split cells

  - maximum size of an item-based CF model is approximately n|I|

  → n|I| * #bytes needed to store an item in a CF model

  → $size_s$: sum of four estimated cell size

  → $size_s$ / ($size_s$ + the size of the parent cell)

Non-Spatial User Ratings for Spatial Items

- Query Processing

  - a single model with travel penalty

  - ranking each spatial item i for a querying user u based on RecScore(u, i)

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

  - P(u, i) = the standard item-based CF predicted rating of item i for user u

  - TravelPenalty(u, i) = road network travel distance between u and i normalized to the same value range as the rating scale

Non-Spatial User Ratings for Spatial Items

- Algorithm2 of Query Processing

  1) KNN algorithm → R with k items with lowest travel penalty

  2) Setting LowestRecScore as the RecScore of the k_{th} item in R

  3) Retrieving items one by one in the order of their penalty score

  4) Calculating the maximum score(MAX_RATING- TravelPenalty(u, i)) for each item

  5) Early termination

  ：If item i cannot make it into the list of top-k recommended items with this maximum possible score

Spatial User Ratings for Spatial Items

- Query processing uses Algorithm 2
- Different P(u,i)

  : using the (localized) collaborative filtering model from the partial pyramid cell that contains the querying user

Experiments

- LARS-T : LARS with only travel penalty enabled
- LARS-U : LARS with only user partitioning enabled
- LARS : LARS with both techniques enabled
- Quality Measure

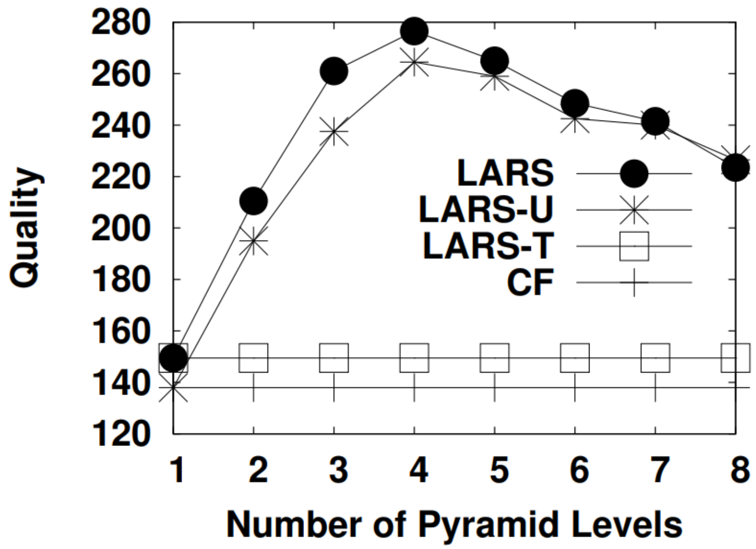
  : R (a set of k recommendations)

  : t (each rating for items known to be liked by user)

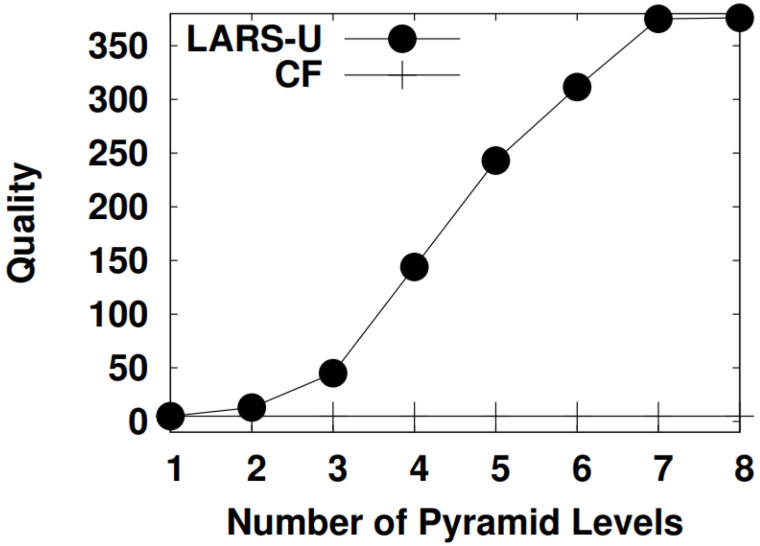
  : the count of how many times R contains the item associated with t

  (the higher the better)

Experiments



(a) Foursquare data

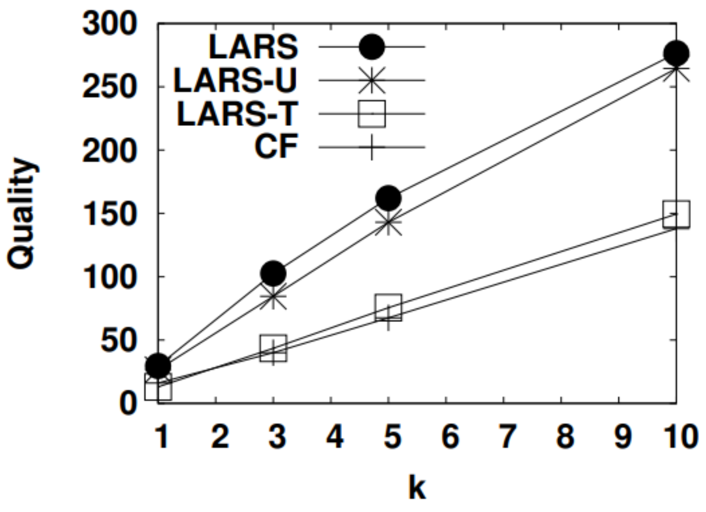


(b) MovieLens data

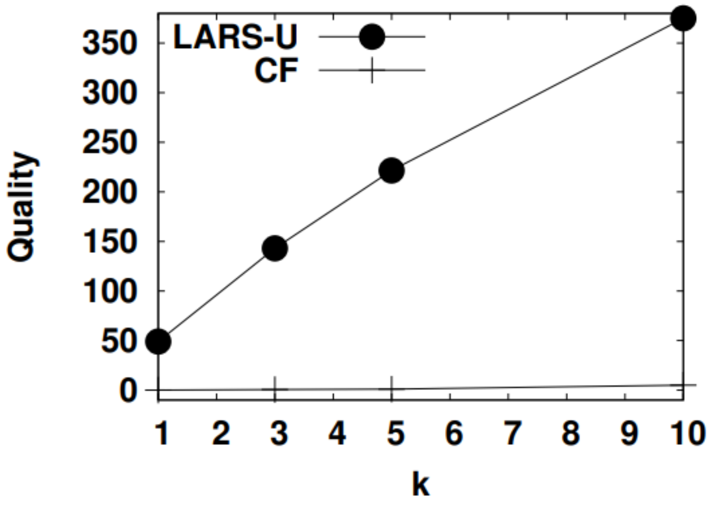
(a) the benefit of using the travel penalty technique that recommends items within a feasible distance

(b) user partitioning is beneficial in providing quality recommendations localized to a querying user location, even when items are not spatial

Experiments



(a) Foursquare data

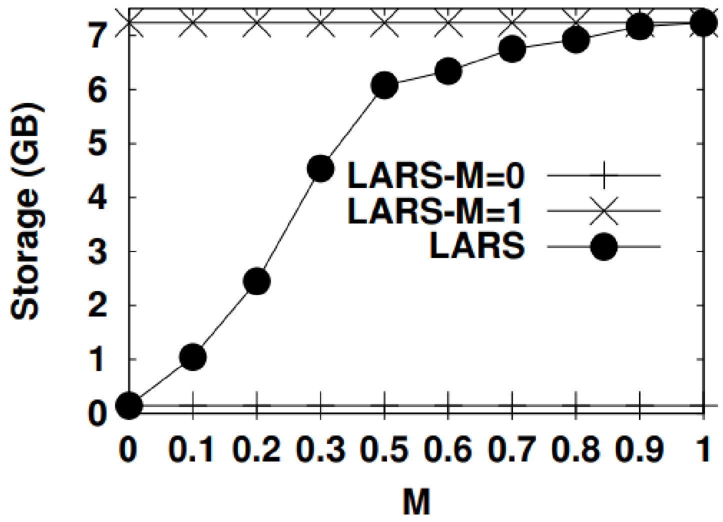


(b) MovieLens data

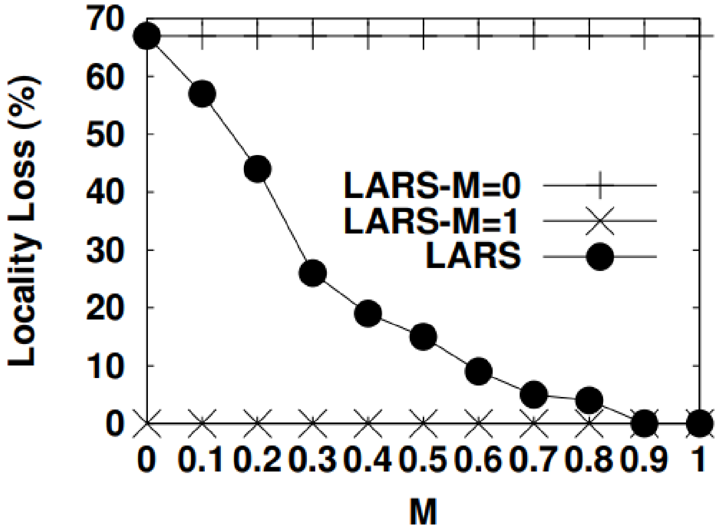
(a) LARS is consistently twice as accurate as CF for all k

(b) LARS-U consistently exhibits better quality than CF for sizes of K from one to ten

Experiments



(a) Storage

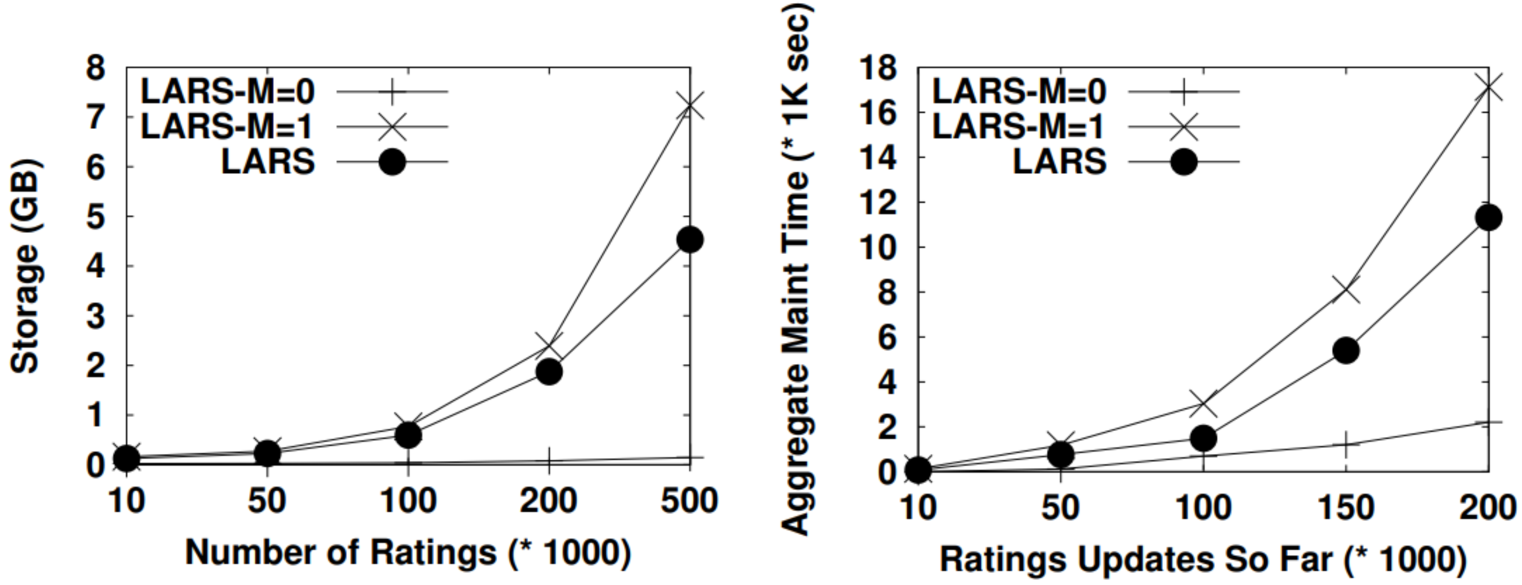


(b) Locality

(a) 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

(b) increasing M results in smaller locality loss as LARS merges less and maintains more localized cells

Experiments

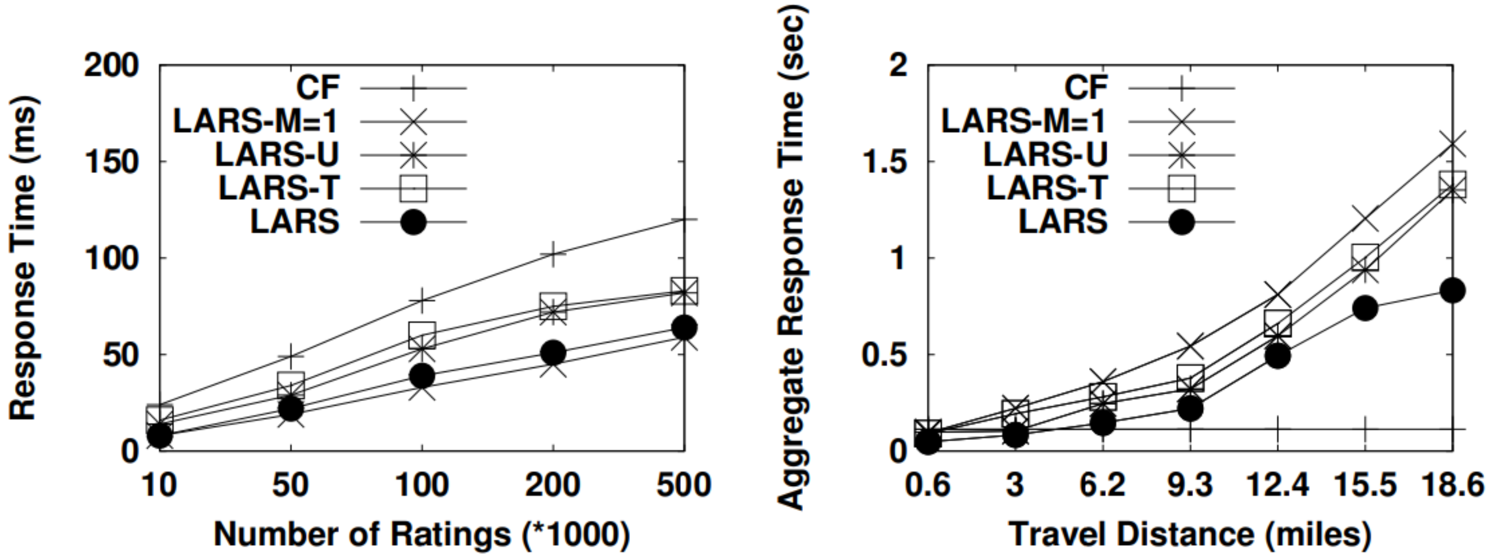


(a) Storage (b) Maintenance

(a) 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

(b) LARS exhibits better performance than LARS-M=1 due to merging

Experiments



(a) Snapshot Queries (b) Continuous Queries

(a) Employing the travel penalty technique with early termination leads to better query response time

(b) LARS exhibits a better aggregate response time since it employs the early termination algorithm using a localized collaborative filtering model to produce results while also merging cells to reduce update frequency

THANK YOU