Online Learning in Large-Scale Recommender System

Amiao Pan
Rudresha Gulaganjihalli Parameshappa

Agenda

A. Introduction
B. Efficient Retrieval of Recommendation
    a. Challenge and Solution
C. Cold-Start Recommendation
    a. Challenge and Solution
D. Conclusion
E. Reference

Introduction

Online Learning Recommender System: is continuous learning system based on the user's behaviour currently to adjust the recommendation for users. It observes how often user accept recommendation.

The Acceptance is most commonly measured by click-through rate (CTR), i.e. the ratio of clicked recommendations. For instance, if a system displays 10,000 recommendations and 120 are clicked, the CTR is 1.2%.

Introduction

Requirement for Online Learning Large-Scale Recommender System

1. The systems are able to recommend the item for user fast
    a. **Scalability Challenges**
2. The systems are able to handle new data that come in continuously.
    a. **Cold Start Challenges**

Agenda

A. Introduction
B. Efficient Retrieval of Recommendation
C. Cold-Start Recommendation
D. Conclusion
E. Reference

Efficient Retrieval of Recommendation

Measurement of Efficient Retrieval: **Retrieval Time of Recommendation.**

Main Challenge:

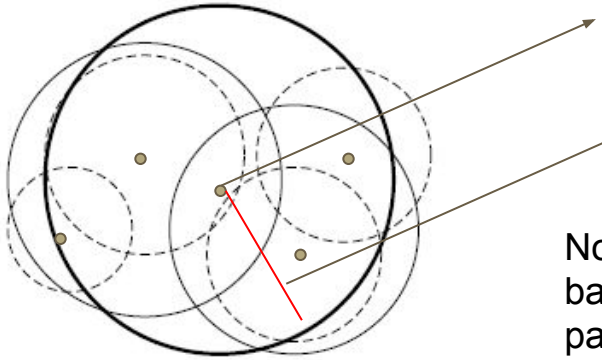
Large scale of User Space, Item Space, Context Space

------reducing retrieval time in Large-Scale Recommender Systems

Approach--Metric Tree[1]

1. Metric Tree used for represent the items space

-----binary space-partitioning trees. The space is partitioned into overlapping hyper-spheres(balls) containing the points



T.center(the mean of the data in that node)

T.radius(the radius of the ball around the mean enclosing the points in thes node)

Note: All the points of child node lie also inside the parent ball, but the child ball itself does not necessarily lie within the parent ball

Fig.1. Metric Tree [1]

Approach--Metric Tree[1]

1. Metric Tree

-----The tree construction is very fast and space efficient.

Fast: the vectors in the items' matrix are sorted during the tree construction.

-----Avoid the random memory access while accessing all the items in the same leaf node.

Space efficient: every node only store the indices of the item vectors instead of the item vectors themselves.

Approach--Metric Tree[1]

1. Metric Tree

Algorithm 4 FindExactRecommendations(User p_u, Item Tree Node Q)

p_u.ub $\leftarrow 0$;
p_u.candidates $\leftarrow \emptyset$;
SearchMetricTree(p_u, Q);
return p_u.candidates;

Fig.2. Algorithm FindExact Recommendations[1]

P_u.candidates: the set of current best K candidate items for user P_u
P_u.ub: the lowest affinity between the user and its current best candidates(items).

Approach--Metric Tree[1]

Performance of Metric Tree

$$\text{Speedup} = \frac{\text{Retrieval Time Using Naive Search}}{\text{Retrieval Time Using Tree}}. \quad [3]$$

	Netflix	MSD	YMusic	Books
Speedup	25.036	21.422	31.219	11.218

Table.1. Speedup Value[2]

Approach--Item Cluster Tree[4]

2.Item Cluster Tree

To solve the scalability problem, we can separately considering :

1. items space
2. Context space

resulting in lower -complexity online learning

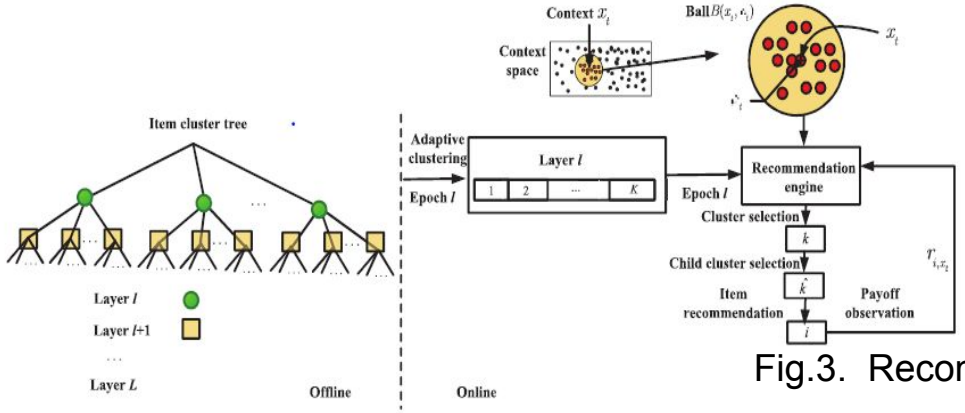
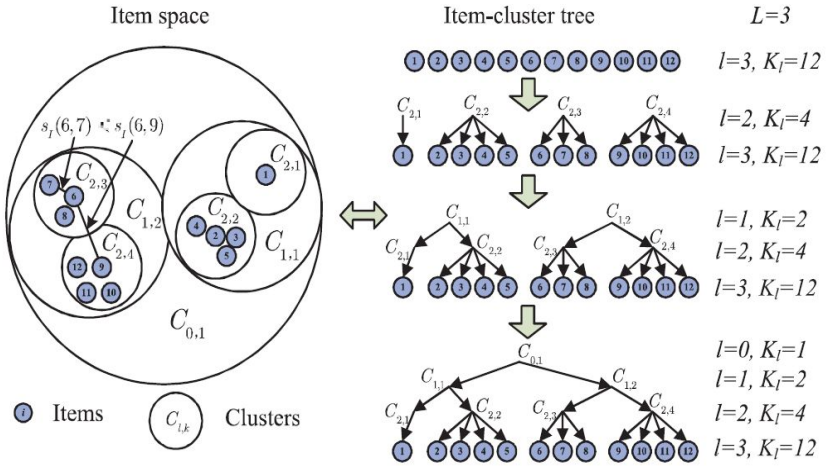


Fig.3. Recommender System on ARC algorithm [4]

Approach--Item Cluster Tree[4]

Since the number of items is large, we cluster them based on the similarity in the item space in order to reduce the possible option for recommendation and then solve the scalability problem.

1. **Cluster size**
2. **Number of Clusters**



Balance the speed of learning and the accuracy by adjusting the number of clusters and the cluster size

Fig.4.:Construction of Item-Cluster Tree [4]

Approach--Item Cluster Tree[4]

b: The b control the trade-off of the number of clusters(learning speed) and the cluster size(accuracy).

$$\begin{bmatrix} b^0 \\ b^1 \\ b^2 \\ \cdots \\ b^l \end{bmatrix}$$

$S_T(\mathrm{l})$: *maximum cluster size at depth* l

$S_T(\mathrm{l})$: , *bounded by* $b^l, b \in (0.5,1)$→exponential tree metric→controlling the Cluster size

Maximum number of clusters at layer l, bounded by $\mathrm{C_I}\left(\frac{1+b}{1-b}\right)^{d_I}\left(\frac{1}{b}\right)^{ld_I}$

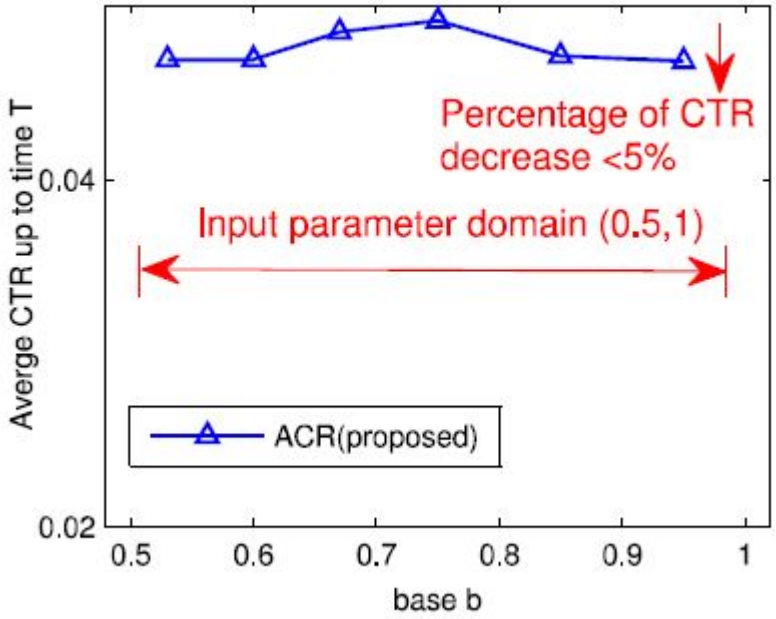
$\mathrm{C_I}$ *and* d_I *are the convering constand and covering dimension for the item space*

l is larger

->cluster size at layer l is smaller

->number of clusters at layer l is larger

Approach--Item Cluster Tree[4]



When b is small, learning is slow but more accuracy,
When b is large, learning is fast but less accuracy.

Optimal b, balance the speed of learning and accuracy.

Fig.5. The Impact of b on the ACR algorithm [4]

Approach--Item Cluster Tree[4]

1. The Item-Cluster Tree is implemented **offline**, the algorithm does not affect the computation complexity of the online recommendation
2. The algorithm make recommendation on **cluster level**, so the increase in items in the system only affects the number of items in a cluster, and does not affect the performance of the algorithm.

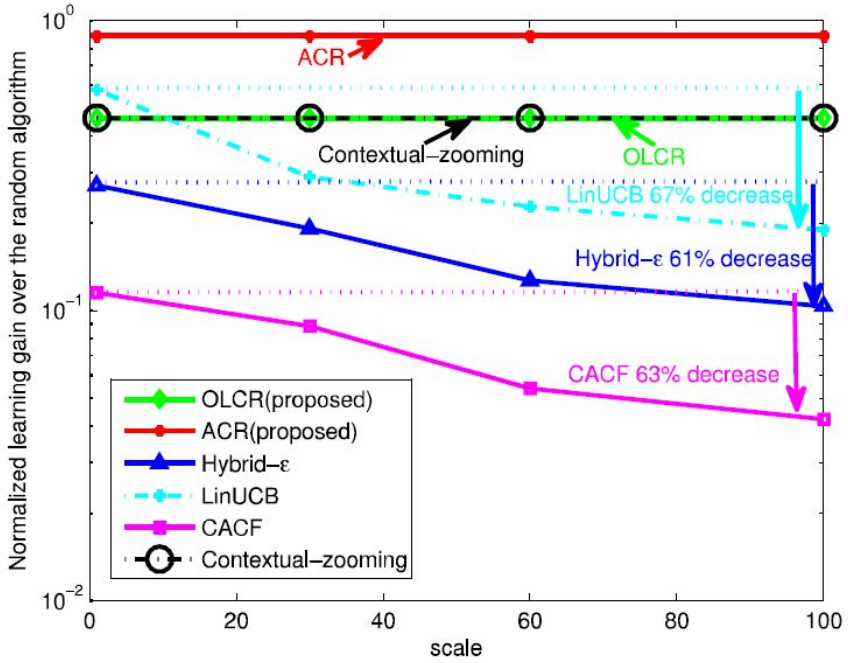


Fig.6: Comparison of scalable performance of algorithm [4]

Approach--Fast Top-k Recommendation [8]

Given a query q=(u,t) , return k items that match u's interests and the temporal context at time t.

Traditional approach: produce the top-k recommendations is to first compute a ranking score for each items and rank all of them. When number of available items becomes large, to produce a top-k ranked list is very time consuming.

Fast Top-k recommendation: an efficient query-processing technique by extending the Threshold Algorithm(TA)[9].

1. Precompute K sorted lists of items. Each list corresponde to be latent topic and the values are sorted based on their generative probabilities with respect to the corresponding topic.

K latent topic: K1 user-oriented (users' personal interest) topics + K2 time-oriented (temporal context) topics

Approach--Fast Top-k Recommendation [8]

ALGORITHM 4: Threshold-based algorithm

Input: A query $q = (u, t)$; inferred model parameters θ_u^t, θ_t', and λ_u; ranked lists $(L_1, \ldots, L_K)$;
Output: List L with all the k highest ranked items;

```
1  Initialize priority lists PQ, L, and the threshold score S_Ta;
2  for z̃ = 1 to K do
3      v = L_z̃.getfirst();
4      Compute S(u, t, v) according to Equation (41);
5      PQ.insert(z̃, S(u, t, v));
6  end
7  Compute S_Ta according to Equation (42);
8  while true do
9      nextListToCheck = PQ.getfirst();
10     PQ.removefirst();
11     v = L_nextListToCheck.getfirst();
12     L_nextListToCheck.removefirst();
13     if v ∉ L then
14         if L.size() < k then
15             L.insert(v, S(u, t, v));
16         end
```

-Compute the ranking score of the first item from every list. It is also the priority of this list. Form Priority List (PL)

-In each iteration, select the highest priority from PL, add it to resulting list L

Approach--Fast Top-k Recommendation [8]

```
17        else
18            v' = L.get(k);
19            if S(u, t, v') > S_Ta then
20                break;
21            end
22            if S(u, t, v') < S(u, t, v) then
23                L.remove(k);
24                L.insert(v, S(u, t, v));
25            end
26        end
27    end
28    if L_nextListToCheck.hasMore() then
29        v = L_nextListToCheck.getfirst();
30        Compute S(u, t, v) according to Equation (41);
31        PQ.insert(nextListToCheck, S(u, t, v));
32        Compute S_Ta according to Equation (42);
33    end
34    else
35        break;
36    end
```

If size(L)> k, compute ranking score of the kth item. If ranking score> Threshold, break; otherwise, replaced by currently v

Update the priority of currently list as well as threshold of each iteration

Approach--Fast Top-k Recommendation [8]

Performance:

The algorithm has the nice property of terminating early without scanning all items.

This TA-based scheme allows us to efficiently return the top-k recommendations by computing the ranking score for the minimum number of items instead of all.

Faster approximate retrieval[1]

1. Clustering users, pre-computing recommendations for the cluster centers and use these recommendations as approximate recommendation for incoming users with similar tastes.

2. Clustering queries, pre-computing solutions for certain representative queries and uses these solutions for the new queries.

These make the retrieval process extremely efficient.

Agenda

A. Introduction
B. Efficient Retrieval of Recommendation
    a. Challenge and Solution
C. Cold-Start Recommendation
    a. Challenge and Solution
D. Conclusion
E. Reference

Cold Start

Cold start is a potential problem in computer-based information systems which involve a degree of automated data modelling. Specifically, it concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information

Cold Start - New User [5]

- Lack of a profile of preferences
  - Not an issue for non-personalized
  - When possible, provide useful default personalization options
    - Popular items, demographically relevant
    - Product association
    - Even trust-based social network data
  - Or get explicit preferences (products, attributes)
  - Phase in personalization as feasible

Fig.7 : New User [6]

Cold Start - New Item [5]

- Challenge: Can’t recommend
- Options
  - Use content-based approaches (including similarity to other items) as an early proxy
  - Recommend to random, or well-chosen set of users (people with diverse tastes, tolerance, interest, influence)

Fig.8 : New Item [6]

Cold Start

Online Learning in large scale Recommender systems

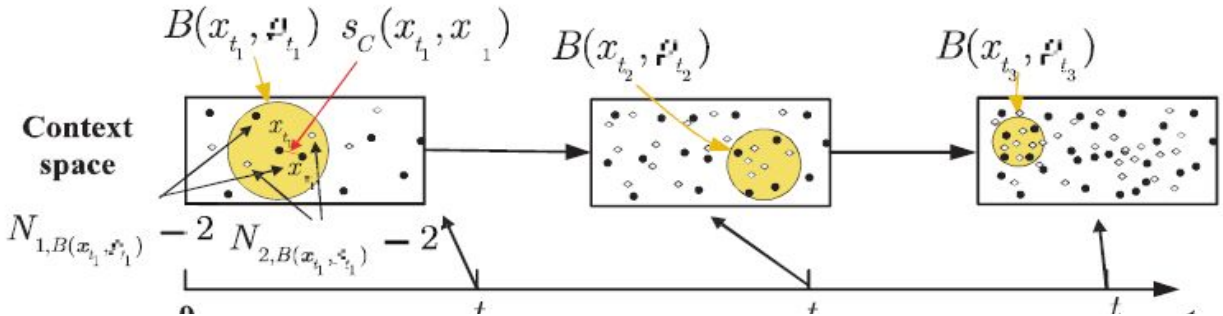


Fig.9: Context arrivals and refinement in the context space [4]

Cold Start

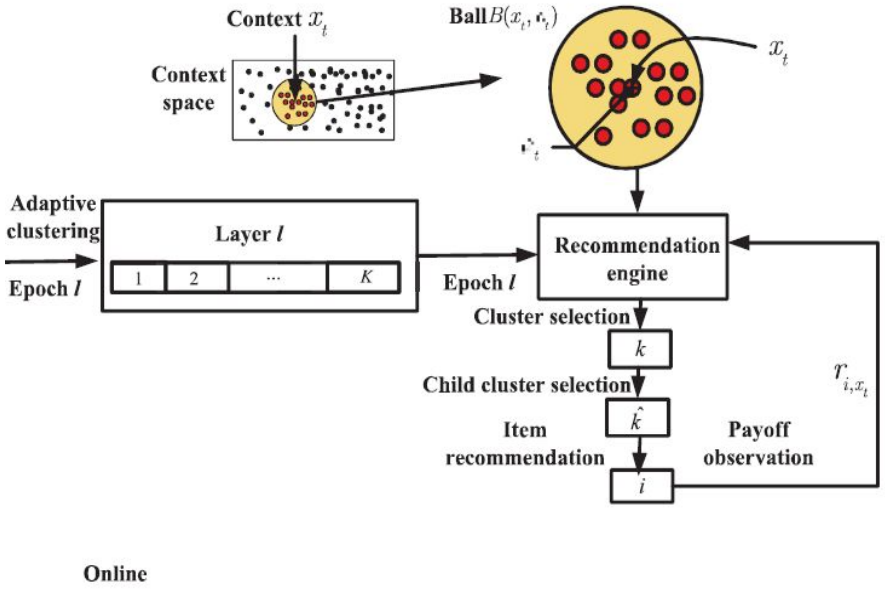


Fig.10 : online calculation in the context space [4]

Cold Start

Experimental Results

		Percentage of user arrivals ($K = 50$)					Percentage of user arrivals ($K = 150$)				
		20%	40%	60%	80%	100%	20%	40%	60%	80%	100%
CTR	CACF	0.028	0.029	0.027	0.029	0.029	0.028	0.029	0.028	0.029	0.029
	Hybrid-ε	0.034	0.035	0.034	0.033	0.033	0.033	0.034	0.032	0.033	0.033
	LinUCB	0.035	0.039	0.040	0.041	0.041	0.032	0.037	0.039	0.039	0.039
	Contextual-zooming	0.037	0.039	0.038	0.038	0.038	0.037	0.039	0.038	0.038	0.038
	OLCR	0.028	0.033	0.037	0.037	0.038	0.025	0.028	0.031	0.032	0.032
	ACR	0.039	0.044	0.046	0.049	0.049	0.039	0.044	0.046	0.049	0.049
	ACR over CACF	39%	52%	70%	69%	69%	39%	52%	64%	69%	69%
	ACR over Hybrid-ε	15%	26%	35%	48%	48%	18%	29%	44%	48%	48%
	ACR over LinUCB	11%	13%	15%	20%	20%	22%	19%	18%	26%	26%
	ACR over contextual-zooming	5%	13%	21%	29%	29%	5%	13%	21%	29%	29%

Fig.11 : Comparison of Click Through Rate of ACR with contextual algorithms [4]

Cold Start

Music Recommender by Modelling Internet Radio Streams

- Problems
  - Vast variety of music available in the internet.
  - Recommending newer music.
  - Recommend to newer user.
  - Continuous and faster learning.

Cold Start

Station Based approach

- Maximum likelihood estimation is used for ranking items
- For faster training use approximated gradient descent along with proposal distribution.
- This approach is station based model [include station while recommending]
- But fails in solving cold start problems

Cold Start

Station Less approach

- Ignore station name while training recommender
- New station issue is solved
- Recommend new user current trending items
- Continuously learn based on user click behavior

Cold Start

Experimental Results

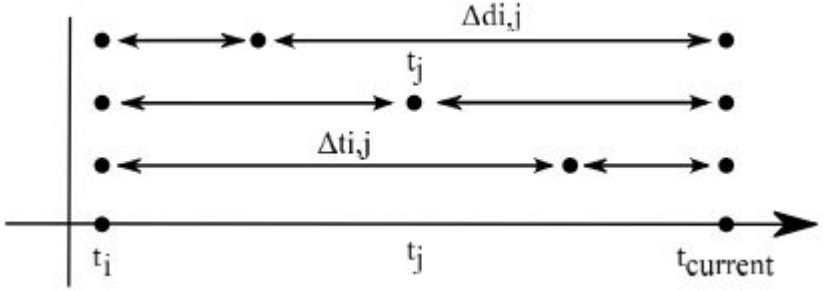
Method	NonRepeat-vs-Uni	NonRepeat-vs-Pop	Played-vs-Pop
Popularity	65.12%	39.91%	50.26%
Genre	78.41%	73.89%	79.33%
k-Means	74.61%	72.83%	83.80%
station-based model	90.91%	87.38%	95.72%
station-less model	**91.41%**	**88.66%**	**95.95%**

Fig.12 : Comparative model accuracy [7]

Cold Start

Time Aware Recommendation system [10]

- Combination of online and offline learning



Then calculate $S_{i,j} = \frac{1}{(\Delta t_{i,i} + \Delta d_{i,i})}$ where $\Delta t_{i,j} = \left| (t_i - t_j) \right|$ and $\Delta d_{i,j} = \left| \min(t_i, t_j) - t_{current} \right|$.

Fig. 13 : Different possibilities of the similarity between two items i and j related to the current date t

Cold Start

Algorithm [10]

Input: A `TimeDataModel` (a list of `TimePreferences` for every user)
Ouput: A ranked list of recommendations for an item
Calculate the list of users UwP having a preference for item i and item j as follows:

$$UwP_{i,j} = \{u \mid p(u,i) \wedge p(u,j)\}$$

$$\forall u \in UwP_{i,j} \text{ calculate } S_{i,j} \text{ and}$$

return a list of the top n items ranked by their averaged similarities $avg(S_{i,j})$.

Fig. 14 : Pseudocode of our time based recommendation algorithm

Cold Start

New User Cold start Problem

- Recommend Top N trending items.
- Do not recommend items already studied by the user.
- Compute Similarity matrix offline
- Do recommendation online

Conclusion

Two challenge:

1. Efficient Retrieval of Recommendation:
    a. metric tree
    b. item-cluster Tree
    c. fast top-k recommendation)
2. Cold Start Recommendation
    a. New user
    b. New item
    c. New system

Reference

[1] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. Efficient retrieval of recommendations in a matrix factorization framework. In Proceedings of the 21st ACM international conference on Information and knowledge management, pages 535{544. ACM, 2012.

[2] Noam Koenigstein and Yehuda Koren. Towards scalable and accurate item-oriented recommendations. In Proceedings of the 7th ACM conference on Recommender systems, pages 419{422. ACM, 2013.

[3] F. P. Preparata and M. I. Shamos. Computational Geometry: An Introduction. Springer, 1985.

[4]L. Song, C. Tekin, and M. van der Schaar. 2016. Online Learning in Large-Scale Contextual Recommender Systems. IEEE Transactions on Services Computing 9, 3 (May 2016), 433–445.

Reference

[5] Nearest Neighbor Collaborative Filtering(Item-Item Collaborative Filtering Recommenders Part 2) - https://www.coursera.org/learn/collaborative-filtering/home/welcome

[6] Mehdi Elahi. Active Learning in Collaborative Filtering Recommender Systems : a Survey. https://www.slideshare.net/MehdiElahi1/active-learning-in-collaborative-filtering-recommender-systems-a-survey

[7] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. Build your own music recommender by modeling internet radio streams. In Proceedings of the 21st international conference on World Wide Web, pages 1–10. ACM, 2012.

[8] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. Dynamic user modeling in social media systems. ACM Transactions on In-formation Systems (TOIS), 33(3):10, 2015.

[9] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. Journal of computer and system sciences, 66(4):614{656, 2003.

[10] Christoph Hermann. Time-Based Recommendations for Lecture Materials. EdMedia: World Conference on Educational Media and Technology, Jun 29, 2010 in Toronto, Canada ISBN 978-1-880094-81-5