Scalable Hierarchical Recommendations Using Spatial Autocorrelation

Ayushi Dalmia
IIIT Hyderabad
Hyderabad, Telengana 500032, India
ayushi.dalmia@research.iiit.ac.in

Joydeep Das
The Heritage Academy
Kolkata, West Bengal 700107, India
joydeep.das@heritageit.edu

Prosenjit Gupta
Dept. of Computer Sc. and Engg.
Heritage Institute of Technology
Kolkata, West Bengal 700107, India
prosenjit_gupta@acm.org

Subhashis Majumder
Dept. of Computer Sc. and Engg.
Heritage Institute of Technology
Kolkata, West Bengal 700107, India
subhashis.majumder@heritageit.edu

Debarshi Dutta
Dept. of Computer Science
University of Southampton, Highfield Campus
Southampton, SO17 1BJ, UK
d30190@gmail.com

ABSTRACT

Collaborative Filtering (CF) is one of the most successful and widely used approaches behind Recommendation Algorithms. CF algorithms use the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users. In this paper, we deal with the *Scalability* issue which is one of the main challenges to the CF algorithms. In Collaborative Filtering, finding similarity amongst N users is an $O(N^2)$ process. If N is large then similarity computation becomes very expensive. In this work, we propose a *Quadtree* based user partitioning technique that partitions the entire users' space into regions based on the location. We develop a Spatially Aware Recommendation Algorithm, where the Recommendation Algorithm is applied separately to each region and therefore allows us to reduce the quadratic complexity associated with the CF process. The proposed work tries to measure the Spatial Autocorrelation indices, such as Geary's index in the regions or cells formed by the Quadtree decomposition. One of the main objectives of our work is to reduce the running time as well as maintain a good quality of recommendation. This approach of recommendation using the decomposition method makes our algorithm feasible to work with large datasets. We have tested our algorithms on the MovieLens and the Book-Crossing datasets.

Categories and Subject Descriptors

I.1.2 [**Algorithms**]; H.3.3 [**Information Search and Retrieval**]: Information Filtering

General Terms

Algorithms, Performance

Keywords

Collaborative Filtering, Spatial Autocorrelation, Quadtrees, Recommendation Algorithms, Scalability

BigDataScience'14, August 04 - 07 2014, Beijing, China

1. INTRODUCTION

In today's world, we are overwhelmed with choices; a plethora of options are available for nearly every aspect of our lives. We need to make choices on a daily basis, from automobiles to home theatre systems, finding Mr. or Ms. Perfect to selecting attorneys or accountants, movies to song, and so on. Today the problem is not about how to get correct information to make a decision, rather, how to make a right decision out of enormous information.

This problem of information overloading has led to the development of information filtering systems and one such subclass is Recommendation Systems or Recommendation Engines. Recommendation Systems typically use the opinions of a community of users (Collaborative Filtering [1, 6, 13]) to help individuals in the community to effectively identify content of interest from a potentially overwhelming set of choices. However the majority of these Recommender Systems focus on recommending the most relevant items to individual users and do not take into consideration any contextual information, such as time, place and company of other people. Note that in many applications it may not be sufficient to consider only users and items, rather it would be necessary to incorporate the contextual information into the recommendation process in order to improve the quality of recommendation. Several context-aware Recommender Systems have been proposed [3, 11, 14] to incorporate contextual information into the existing Recommendation Algorithms. However, the existing context-aware Recommender

Systems cannot efficiently combine different types of contextual information, and also suffer from high computational complexity. In this paper, we propose a location aware Recommendation Algorithm which considers both the location of the target user and the preferences of the other users within the same region in the recommendation process. We partition the entire users' space with respect to location by using a Quadtree [1] based data structure. The primary objective of our work is to decompose the users' space into regions in such a way that the correlated users (users having similar preferences over items) end up in the same regions. Spatial Autocorrelation [9] index (Geary's index) is applied to measure the correlation among the users in the regions resulting from space decomposition. Quadtree partitions the entire space into smaller regions (cells), and we offer recommendations to the users in those cells independently.

For recommendation generation, Collaborative Filtering algorithm is applied to the different regions of the tree. One of the motivations of our work is to cope with the quadratic complexity typically associated with Collaborative Filtering algorithms. In Collaborative Filtering, finding similarity amongst N users is an $O(N^2)$ process. If N is large then similarity computation becomes quite expensive. Decomposition avoids this quadratic blowup and allows us to process bigger data sets even with limited computational resources. As for example, if we partition a region with n users into k partitions with nearly equal sizes, then the overall time required for performing collaborative filtering in all those k partitions will be proportional to $k.(n/k)^2 = (n^2/k^2).k = n^2/k$. So we can achieve a k order speed up by dividing the users' space into k partitions. Note that though we decomposed the users' space into smaller regions and applied the Recommendation Algorithm to the regions separately, it does not mean that two distant users cannot have high correlation in rating. Our goal is to partition the space into smaller manageable regions and in turn reduce the overall running time without sacrificing recommendation quality. This ensures scalability, allowing us to tackle bigger datasets. Experiments conducted indicate that our method is effective in reducing the running time, while maintaining an acceptable quality of recommendation.

2. PRELIMINARIES

2.1 Quadtree

A Quadtree is a tree data structure in which each internal node has exactly four children. Quadtrees are used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions. The regions may be square or rectangular, or may have arbitrary shapes. All forms of Quadtrees share some common features:

- They decompose space into adaptable cells.
- Each cell (or bucket) has a maximum capacity. When maximum capacity is reached, the bucket splits.
- The tree directory follows the spatial decomposition of the Quadtree.

2.2 Spatial Autocorrelation

[1]http://en.wikipedia.org/wiki/Quadtree

Spatial data analysis shows that in geographic space nearby things are more related than distant things. Spatial data tends to be highly self-correlated. In the same neighborhood, people with similar characteristics, occupations and backgrounds tend to cluster. The analysis of spatial data in spatial statistics is called Spatial Autocorrelation. Spatial Autocorrelation [9] measures the co-variance of properties within geographic space and it deals with both attributes and locations of spatial features. A commonly used measure of Spatial Autocorrelation is Geary's index (c). Geary's Index measures the similarity of i's and j's attributes, c_{ij}, which can be calculated as follows:

$$c_{ij} = (z_i - z_j)^2$$

where z_i and z_j are the values of the attribute of interest for objects i and j.

A locational similarity w_{ij} is used in the calculation of Geary's index, where $w_{ij} = 1$ if i and j share a common boundary, and $w_{ij} = 0$ if not. Geary's index is expressed as follows:

$$c = \frac{\sum_i \sum_j w_{ij} c_{ij}}{2 \sum_i \sum_j w_{ij} \sigma^2}$$

where σ^2 is the variance of the attribute z values, or

$$\sigma^2 = \frac{\sum_i (z_i - \bar{z})^2}{(n-1)}$$

and

$$\bar{z} = \frac{\sum_i^n z_i}{n}$$

If $c = 1$, the attributes are distributed independently of location. If $c < 1$, similar attributes coincide with similar locations, and if $c > 1$, attributes and locations are dissimilar.

2.3 Collaborative Filtering

Collaborative Filtering (CF) is one of the most commonly and widely used techniques for developing Recommendation Engines. Collaborative Filtering, also known as Social Information Filtering, is based on the principle of finding a subset of users who have similar taste and preferences to that of the active user, and offering recommendations to the active user based on that subset of users. The idea is that given an active user u, compute her n similar users $\{u_1, u_2, \cdots, u_n\}$ and predict u's preference based on the preferences of $\{u_1, u_2, \cdots, u_n\}$. Similar users mean users who share the same kind of tastes and preferences over items. The basic idea behind Collaborative Filtering is that users who agreed on the past tend to agree in the future also. Collaborative Filtering process typically uses a large database of preferences for items by users to predict additional items or products a new user might like.

3. PAST WORK

A lot of work has been done in the area of Recommendation Engines in the past. Examples of online Recommendation Systems in the movie domain include MovieLens[2],

[2]http://www.movielens.org/

Netflix[3]. They used Collaborative Filtering approach to recommend movies. Li et al. [8] integrated GPS into Recommender System to create a location-aware Recommender System. They explored the increasing demand of mobile commerce and developed a Recommender System for mobile commerce in tourism. Adomavicius and Tuzhilin [2] incorporated the relevant contextual features that may effect the preferences of the user into their Recommendation Algorithm. They developed a Context Aware Recommender System that uses both user preferences as well as the current context in their recommendation process. Levandoski et al. [7] used location based rating to produce recommendations. Their work, LARS, classified three novel classes of location based ratings, namely, spatial ratings for non-spatial items, non-spatial ratings for spatial items, and spatial ratings for spatial items. Das et al. [5] used Voronoi Diagrams to tessellate the plane and applied the Recommendation Algorithm separately in each Voronoi cell. Sarwar et al. [12] clustered the complete user set on the basis of user-user similarity and used the cluster as the neighborhood. O'Conner et al. [10] use clustering algorithms to partition the item set on the basis of user rating data.

4. OUR CONTRIBUTION

The motivation of our work comes from the property of *Preference Neighborhood* often found in spatial data analysis. The term *Preference Neighborhood* suggests that users from a spatial region (e.g., locality) prefer items (e.g., movies, destinations) that are noticeably different from items preferred by users from other, even adjacent regions. Preference Neighborhood suggests that our recommendations should be influenced by rating data history spatially close to the user. The above described property motivated us to design a Recommendation Algorithm which will include user location as one of the parameters. In this work, we propose a spatially-aware Recommender System using Quadtree based space partitioning techniques. As mentioned in Section 1, location also allows us to decompose the problem space thereby allowing us to potentially scale up to larger datasets.

Our system, while recommending, tries to find out the location of the user who will receive the recommendation, and then explores the preferences of his neighborhood using Collaborative Filtering to generate the recommended items. We have developed two such spatially aware engines, one for books and another for movies. We have used the Book-Crossing dataset to test our algorithm. Book-Crossing dataset[4] contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit/implicit) on 271,379 books. Ratings are either explicit, expressed on an integral scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0. For the movie recommender system, we have used MovieLens dataset to test our algorithm. MovieLens is a Collaborative Filtering recommender system developed by GroupLens Research Group. The dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. Ratings are on a five star (integral) scale from 1 to 5.

While recommending, our primary focus is on the location (spatial aspect) of the user. We use longitude and latitude to identify the user's location. Our main task is to partition the entire users' space into smaller regions in such a way that the correlated users are placed in the same regions. Quadtree is used to dissect the users' space with respect to location and recommendations will be provided to users in those regions. We use Geary's index to measure correlation among the users in the regions resulting from Quadtree decomposition and then recommend items with the idea that the suggested items will be liked by the user. To offer recommendations to a user in a particular region, our system uses the preferences of the target user and the preference history of the other users of that region. The main idea is to apply the CF algorithm separately to each region, and in turn reduce the overall running time. Pearson's correlation coefficient is used to compute the similarity among the users. In this work, we try to recommend items by computing user-user similarity among the users in the regions.

Table 1: Splitting Criteria

No. of users in the region	Correlation value	Look ahead (one level)	Split
High	High	N/A	Y
High	Low	N/A	Y
Low	High	N/A	N
Low	Low	positive	Y
		negative	N

5. THE DECOMPOSITION ALGORITHM

In this work, we use a Quadtree based approach for space decomposition. Space partitioning is done on the basis of the cities of the users. The Book-Crossing dataset has information about the users and their locations. User location is represented as a triplet $\{city, state, country\}$. In this work, we use the city attribute to tessellate the space, and in the process construct the tree. Similarly for the MovieLens datasets, the zip-codes of the users are mapped to the corresponding latitude and longitude to obtain the user coordinates. To build the tree nodes (partitions), we represent each city as a 2D coordinate by their respective longitudes and latitudes. Our algorithm calculates Geary's index value for each region in the following manner. We use book (or movie) ratings as the parameter for computing attribute similarity (c_{ij}), while locational similarity (w_{ij}) is measured by computing the distance between the cities of a pair of users and use the inverse of the distance to compute similarity (as discussed in section 2.2). If the index value calculated lies within a predefined range, we can infer that spatial autocorrelation exists in the data and the users share similar tastes and preferences. Distances between cities are measured using Haversine formula[5], that computes great-circle distances between two points on a sphere from their longitudes and latitudes.

The space partitioning algorithm first finds the spatial correlation index value for a region, and then applies some splitting criteria (as summarized in Table 1) to split the region into four regions. In our work, the decomposition algorithm initially finds the spatial correlation value for the entire region (level-0 of the tree), and then applying the splitting criteria splits the region into four regions (level-1). At level-1 of the tree, we have four regions, and again the

[3]http://www.netflix.com/

[4]http://www.informatik.uni-freiburg.de/cziegler/BX/

[5]http://www.movable-type.co.uk/scripts/latlong.htm

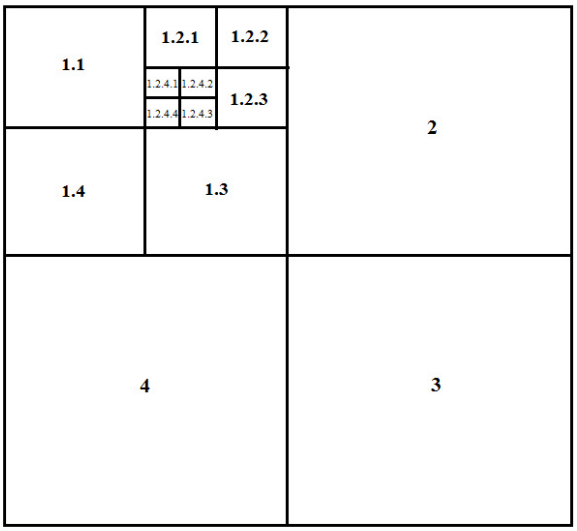

Figure 1: Division of the sample space into regions by recursively subdividing it into four sub-regions

decomposition algorithm is applied to each of the regions individually to split each of them into four different regions (level-2). This process is continued as long as the splitting criterion is satisfied. Figure 1 shows a typical example of this splitting process.

According to Table 1, if the number of users in a region is high (above some threshold), then we always split the region irrespective of the correlation value of the region. If the number of users in a region is low (within a range) and the correlation value is high, then we do not split the region. However, if the number of users is low (within a range) and correlation is also low, we have implemented a concept of one level look ahead. Here, we split the region into four sub-regions, and check the correlation value of each of these sub-regions. If the correlation value of these sub-regions is better than the correlation value of the previous region (positive case), then we proceed with the split, otherwise (negative case) we cancel the split.
The scheme is detailed in its algorithmic form as follows:

Algorithm Quadtree_Decomposition
Step 1: Represent user location (city) as coordinates (longitude-latitude).
Step 2: Find the spatial autocorrelation value of the entire region (level-0 of the tree).
Step 3: Build the tree using splitting criteria.
Step 3.1: If correlation is good and number of users in the region is low (below the threshold limit), we do not split the region.
Step 3.2: If the number of users in a region is high (above the threshold limit), then irrespective of the correlation value we split the region.
Step 3.3: If both the number of users and correlation value of a region is low (below threshold limit), we apply the *look ahead* criteria, and consequently split(or do not split).
Step 4: Repeat steps 3 and 4 for each of the regions (tree nodes) as long as the splitting criterion is met.

6. THE RECOMMENDATION ALGORITHM

The Recommendation Algorithm recommends a list of items to the users in the regions with the idea that the recommended items will be liked by the users. We use Collaborative Filtering algorithms for recommendation and while recommending, the algorithm considers the rating statistics of only the neighboring users of the target user. We emphasize on applying the CF algorithm separately to each region, and in turn reduce the running time. As we will discuss in Section 7, spatial autocorrelation exists in the regions, and hence it seems very likely that if we recommend items based on the preferences of the users who share the same neighborhood with the target user, the quality of the recommendations will improve. For a user seeking recommendation (target user), we identify the location (longitude and latitude) of the user and accordingly map the user to his or her destined region. We find the most likely collaborative users for the target user by calculating Pearson's coefficient value [4] between the target user and all other users in the region. The algorithm finds the *top-10* users in the region who have the highest correlation of ratings with the target user. We then choose 5 of the top rated items (books or movies) from each of these 10 users to form a set of 50 items. For recommendation, 10 of these 50 items are chosen depending upon the local average rating of these items. The algorithm is briefly described below:

Algorithm Recommend_Item
Step 1: Select a user for recommendation.
Step 2: Identify the location (longitude and latitude) of the user.
Step 3: Map the user in the exact region (node) of the Quadtree according to his/her location.
Step 4: Find a subset of users in the region who share similar preferences for items with the active user. Select *top-10* similar users.
Step 5: Select top 5 highly rated item from each of these *top-10* users to from a *top_set* of 50 items.
Step 6: Recommend *top-10* items from the *top_set* by averaging the rating of the items in the region.

7. EXPERIMENTS AND RESULTS

7.1 Pre-processing

The datasets (Book-Crossing and MovieLens) we used have information about the items (books or movies) and their ratings by different users across the world. We kept our dataset confined to only the US users. Since the volume of the dataset is huge and in our work, location is a very important aspect, we decided to remove the users of other locations and in the process we are left with a comprehensive set of users. In our work, we represented the cities by their corresponding longitudes and latitudes. The dataset was pruned by removing users who rated relatively less number of items, and also removing the items that were rated by relatively less number of users.

7.2 Experimentation with Decomposition algorithm

We have tested our algorithm on the Book-Crossing dataset and the MovieLens Dataset to validate our scheme. We have run the decomposition algorithm on the entire preprocessed dataset. The experiment has been performed with the following different threshold parameters: *User Threshold*(n_1

and n_2), *Correlation Threshold*(CT) and *Item Threshold*(f). n_1 and n_2 represent the minimum and maximum number of users in a region respectively, CT is the minimum correlation value a region must have for not being decomposed further and f is the fraction of the total number of items used for correlation calculation. Table 2 lists the decomposition rule, which is as follows. If the number of users in a region is more than n_2 then we perform a split irrespective of the correlation value. If the number of users is within the range then we find the correlation value of the regions after the split, and if it is found to be better, we go ahead with the split or else we roll back to the previous state as discussed in section 5. We have carried out the experiment using different values for n_1, n_2, CT and f. It is found that the decomposition algorithm gives better results when the fraction of items (books or movies) is less. It is so because the items which are not rated by a good number of users do not have much impact on the calculations and do not depict the true parameters of the location. Taking them into consideration implies doing calculations over a less relevant data set and hence reduces the efficiency of our algorithm.

We report the results of experiments performed on the Book-Crossing Dataset in Tables 3 to 5, and that of the MovieLens Dataset in Tables 6 to 8. For example, in Table 3, we can see that region 1 (entire users' space) is decomposed into 4 sub-regions 1.1, 1.2, 1.3 and 1.4 after applying the threshold criteria. Region 1.2 is divided into 1.2.1, 1.2.2, 1.2.3 and 1.2.4 as the number of users (n) in the region is greater than n_2. Similarly Region 1.2.4 is further split into the regions 1.2.4.1, 1.2.4.2, 1.2.4.3 and 1.2.4.4. However Region 1.4 is not divided further as the correlation value of the region using 1/8 of the books is less than 0.5, since $f = 1/8$ and $CT = 0.5$ (see Figure 1). In the Tables, we also reported the Geary's index (Spatial autocorrelation) values for all the regions. Spatial Autocorrelation is calculated using the items rated by the users in the region. We know that if the Geary's index value is less than 1, then spatial correlation is present, else absent. So we define a correlation range $\{< 0.75, [0.75 - 1.0), >= 1.0\}$ for the items and reported the total number of items (movies or books) that fall within each range. Regions with 0 number of users are omitted from the table. The last two columns of the tables show the percentages of items that fall below the correlation value 0.75 and 1.0 respectively. As for example, in Table 3, we notice that on an average 99.07% of the books across all the regions fall below correlation value 0.75 while 99.33% of the books fall below 1.0. In Table 4 we report similar results with different set of threshold parameter. Analyzing both the tables, we can observe that substantial Spatial Autocorrelation exists in the regions, and hence we can conclude that users in a region are highly correlated. We have 2 more Tables for two sets of threshold parameter, but for brevity we only present the summary of those Tables in Table 5. In Table 5, we have presented a comprehensive summary of the decomposition results for the Book-Crossing Dataset. Here we have shown the average correlation values across all the regions using different threshold values. Similarly, we present the results of our experiment with MovieLens Dataset in Tables 6 and 7 for different threshold parameter, and the corresponding summary in Table 8. From Tables 5 and 8, we can observe that the decomposition algorithm produces better results (in terms of correlation values) when the fraction of items (f) is less.

Table 2: Decomposition Rule

Fraction of Items = f		
No. of Users (n)	Correlation	Decomposition Decision
$n > n_2$	Ignore	Split anyways
$n_1 < n \leq n_2$	$< CT$	Do not split
$n_1 < n \leq n_2$	$\geq CT$	split
$n < n_1$	Ignore	Do not split

Table 3: Spatial Decomposition [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.125$]

Correlation Range		< 0.75	$[0.75 - 1.0)$	≥ 1.0	$\% < 0.75$	$\% < 1.0$
Region No.	Total users	No. of books	No. of books	No. of books		
1.1	1594	2535	7	10	99.33	99.61
1.2	5265	3352	6	8	99.58	99.76
1.3	857	1725	3	6	99.48	99.65
1.4	1935	2601	5	10	99.43	99.62
1.2.1	222	806	2	10	98.53	98.78
1.2.2	353	1022	4	5	99.13	99.52
1.2.3	1615	2125	8	9	99.20	99.57
1.2.4	3075	2539	12	6	99.30	99.77
1.2.4.1	653	3117	4	10	99.55	99.68
1.2.4.2	125	463	1	14	96.86	97.07
1.2.4.3	759	1729	4	10	99.20	99.43
1.2.4.4	1538	2463	7	10	99.31	99.60
			Average		**99.07**	**99.33**

Table 4: Spatial Decomposition [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 5000$; $CT = 0.5$; $f = 0.125$]

Correlation Range		< 0.75	$[0.75 - 1.0)$	≥ 1.0	$\% < 0.75$	$\% < 1.0$
Region No.	Total users	No. of books	No. of books	No. of books		
1.1	1594	2535	7	10	99.33	99.33
1.2	5265	3352	6	8	99.58	99.76
1.3	857	1725	3	6	99.48	99.65
1.4	1935	2601	5	10	99.43	99.62
1.2.1	222	806	2	10	98.53	98.78
1.2.2	353	1022	4	5	99.13	99.52
1.2.3	1615	2539	12	6	99.30	99.77
1.2.4	3075	3117	4	10	99.55	99.68
			Average		**99.29**	**99.55**

Table 5: Summary of Spatial Decomposition with various threshold values [*BookCrossing Data*]

n_1	n_2	CT	f	No. of Regions	$\% < 0.75$ (Average)	$\% < 1.0$ (Average)
1000	3000	0.5	0.25	12	98.65	98.87
1000	3000	0.5	0.125	12	99.07	99.33
1000	5000	0.5	0.25	8	99.03	99.22
1000	5000	0.5	0.125	8	99.29	99.55

Table 6: Spatial Decomposition [*MovieLens Data*] [$n_1 = 500$; $n_2 = 1000$; $CT = 0.5$; $f = 0.25$]

Correlation Range		< 0.75	$[0.75 - 1.0)$	≥ 1.0	$\% < 0.75$	$\% < 1.0$
Region No.	Total users	No. of movies	No. of movies	No. of movies		
1.1	2844	257	295	458	25.45	54.65
1.2	1069	251	339	437	24.44	57.45
1.3	1283	261	271	448	26.63	54.29
1.4	319	230	231	431	25.78	51.68
1.1.1	248	193	167	406	25.20	47.00
1.1.2	379	232	198	370	29.00	53.75
1.1.3	1884	233	250	413	26.00	53.91
1.1.4	333	205	250	423	23.35	51.82
1.2.2	18	212	125	264	35.27	56.07
1.2.3	1051	213	242	356	26.26	56.10
1.3.1	808	253	230	387	29.08	55.52
1.3.2	475	298	225	350	34.13	59.9
1.1.3.1	117	278	95	385	36.82	49.01
1.1.3.2	192	268	145	348	35.22	54.27
1.1.3.3	961	253	236	375	29.28	56.60
1.1.3.4	614	266	263	348	30.33	60.32
1.2.3.1	501	300	224	393	32.72	57.14
1.2.3.2	33	266	157	379	33.17	52.74
1.2.3.4	517	285	232	422	30.35	55.06
			Average		**29.38**	**54.59**

Table 7: Spatial Decomposition [*MovieLens Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.125$]

Correlation Range		< 0.75	$[0.75 - 1.0)$	≥ 1.0	$\% < 0.75$	$\% < 1.0$
Region No.	Total users	No. of movies	No. of movies	No. of movies		
1.1	2844	141	157	262	25.18	53.21
1.2	1069	125	116	253	25.30	48.79
1.3	1283	184	144	274	30.56	54.49
1.4	319	173	169	258	28.83	57.00
			Average		**27.46**	**53.37**

Table 8: Summary of Spatial Decomposition with various threshold values [*MovieLens Data*]

n_1	n_2	CT	f	No. of Regions	$\% < 0.75$ (Average)	$\% < 1.0$ (Average)
500	1000	0.5	0.25	19	29.38	54.59
500	1000	0.5	0.125	19	31.64	50.87
1000	3000	0.5	0.25	4	22.38	49.99
1000	3000	0.5	0.125	4	27.46	53.37

7.3 Experimentation with Recommendation algorithm

While recommending items to a user, our algorithm considers the $\{user, item, rating\}$ triplets of only the collaborative users of the target user. As we have seen in the previous section that Spatial Autocorrelation exists in the regions, and therefore it seems very promising that if you recommended only using the users in the region of the target user, recommendation quality will improve. For testing our algorithm, we randomly split the user ratings into two sets - observed items (80%) and held-out items (20%). Ratings for the held-out items were to be predicted. Pearson's Correlation coefficient [4] is used as the similarity metric for finding the collaborative users of a target user. To test the accuracy of the algorithm, we followed a scheme: we executed the algorithm only with those items that are rated by both the target user and her collaborative users. The ratings of the top-10 recommended items produced by the algorithm were then compared with the actual ratings of those items given by the target user to verify whether these items were also highly rated by the target user.

Evaluation metric: Two commonly used metrics for evaluating the prediction accuracy of traditional collaborative filtering algorithms are Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [4]. We use MAE and RMSE to evaluate the prediction accuracy while recommendation quality is measured using the Recall metric [4].

MAE: MAE is defined as the average of the absolute error. Absolute error is the difference between the predicted rating and actual rating. Let the actual user ratings be, $\{r_1, r_2, \cdots, r_n\}$, and predicted ratings are, $\{p_1, p_2, \cdots, p_n\}$, where n is the number of items. Then absolute error,

$$E = \{e_1, e_2, \cdots, e_n\} = \{p_1 - r_1, p_2 - r_2, \cdots, p_n - r_n\}$$

and,

$$MAE = \frac{\sum_{i=1}^{n} |e|}{n}$$

Any prediction algorithm tries to minimize the MAE.

RMSE: RMSE is similar to MAE and is biased to provide more weights to larger errors.

Table 9: Possible Recommendations

	Customer Likes (rating = 4 or 5)	Customer Dislikes (rating = 1, 2 or 3)
Recommend	True positives	False positives
Do not recommend	False negatives	True negatives

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}}$$

Recall: The Recall metric is also known as the hit rate, which is widely used for evaluating top-k recommender systems. Table 9 shows the different combinations of recommendations that can be generated in a recommendation problem. We consider that a customer likes an item (movie or book) if he has given a rating of 4 or 5 to that item (in a scale of 1 to 5), otherwise dislikes it, i.e., his rating is 1, 2 or 3. A recommendation is positive if recommended rating coincides with the actual rating given by the customer. In our Recommender System, Recall measures what fraction of the items liked by the customers, has been recommended by the algorithm.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

To evaluate the quality of overall recommendation, we tested the algorithm for all the regions at different levels of the tree. We report the results of the Recommendation Algorithm performed on the Book-Crossing Dataset in Tables 10 to 14, and that of the MovieLens Dataset in Tables 15 to 19. In the Tables, the column *Time* shows the running time of our Recommendation Algorithm in the different regions, while column *Cum. Time* reports the cumulative running time for the sub-regions belonging to a single region. Our experiments are run on a computer with Core i3 - 2100 @ 3.10GHz x 4 CPU and 2 GB RAM.

Table 10: Results of Recommendation [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.25$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	1594	0.7613	0.8379	0.9225	111.5	
1.2	5265	0.8123	0.9834	0.9444	**3221.33**	
1.3	857	0.6776	0.7381	0.9346	10.78	3458.22
1.4	1935	0.7806	0.8786	0.9599	114.61	
1.2.1	222	0.8822	0.9929	0.9953	0.21	
1.2.2	353	0.7884	0.8961	0.997	0.62	
1.2.3	1615	0.8023	0.9103	0.9515	79.75	**665.27**
1.2.4	3075	0.7156	0.8007	0.9297	**584.69**	
1.2.4.1	653	0.6304	0.6944	0.944	6.33	
1.2.4.2	125	0.7514	0.7929	0.9833	0.5	
1.2.4.3	759	0.7962	0.8846	0.9358	8.28	**96.76**
1.2.4.4	1538	0.7598	0.8425	0.9377	81.65	
	Average	**0.7632**	**0.8544**	**0.953**		

In the Tables, we have reported the total running time (in minutes) of our Recommendation Algorithm for recommending all the users of a region, as well as a cumulative running time for comparing the performance of the algorithm (in terms of running time) in the sub-regions with that of the parent region. As for example, in Table 10, we can notice that the time required for recommending all the users in region 1.2 is 3221.33 minutes. However, when it

Table 11: Results of Recommendation [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.125$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	1594	0.7514	0.8756	0.926	98.54	
1.2	5265	0.8312	0.1.022	0.9259	**2987.43**	
1.3	857	0.6823	0.9234	0.9415	9.78	3199.45
1.4	1935	0.7623	0.9123	0.8996	103.7	
1.2.1	222	0.8812	1.134	0.8571	0.14	
1.2.2	353	0.7882	0.8765	0.9091	0.55	
1.2.3	1615	0.8143	0.9232	0.911	68.38	**584.67**
1.2.4	3075	0.7123	0.8234	0.9297	**515.6**	
1.2.4.1	653	0.653	0.727	0.944	4.5	
1.2.4.2	125	0.7512	0.8123	0.8939	0.4	
1.2.4.3	759	0.7523	0.8923	0.9343	7.3	**86.9**
1.2.4.4	1538	0.7645	0.8534	0.9383	74.7	
	Average	**0.762**	**0.898**	**0.9175**		

Table 12: Results of Recommendation [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 5000$; $CT = 0.5$; $f = 0.25$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	1594	0.7613	0.8379	0.9225	111.5	
1.2	5265	0.8123	0.9834	0.9444	**3221.33**	
1.3	857	0.6776	0.7381	0.9346	10.78	3458.22
1.4	1935	0.7806	0.8786	0.9599	114.61	
1.2.1	222	0.8822	0.9929	0.9953	0.21	
1.2.2	353	0.7884	0.8961	0.997	0.62	
1.2.3	1615	0.8023	0.9103	0.9515	79.75	**665.27**
1.2.4	3075	0.7156	0.8007	0.9297	584.69	
	Average	**0.7775**	**0.8797**	**0.954**		

Table 13: Results of Recommendation [*BookCrossing Data*] [$n_1 = 1000$; $n_2 = 5000$; $CT = 0.5$; $f = 0.125$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	1594	0.7514	0.8756	0.926	98.54	
1.2	5265	0.8312	0.1.022	0.9259	**2987.43**	
1.3	857	0.6823	0.9234	0.9415	9.78	3199.45
1.4	1935	0.7623	0.9123	0.8996	103.7	
1.2.1	222	0.8812	1.134	0.8571	0.14	
1.2.2	353	0.7882	0.8765	0.9091	0.55	
1.2.3	1615	0.8143	0.9232	0.911	68.38	**584.67**
1.2.4	3075	0.7123	0.8234	0.9297	515.6	
	Average	**0.7779**	**0.9363**	**0.9125**		

is decomposed into four sub-regions 1.2.1, 1.2.2, 1.2.3, and 1.2.4, we have a total running time of 665.27 minutes for all the sub-regions, which is significantly less (by about 80%) than the time required for recommending using the entire region 1.2. Similarly the running time for region 1.2.4 is 584.69 minutes, while the cumulative running time for all of its sub-regions is 96.76 minutes (reduced by about 84%). We also reported similar comparisons in the remaining Tables. It is clear from these Tables that the running time of the algorithm is reduced when it is applied to the sub-regions separately than the entire region. Note that the regions with 0 number of users are omitted from the Tables.

We use MAE, RMSE, and Recall metrics to evaluate our Recommendation Algorithm. A Recall value of 1.0 indicates that the algorithm is always able to recommend the hidden items. However, a high Recall alone does not indicate good recommendations, as by recommending the full set of items available, one can trivially generate a Recall value of 1.0. In the Tables, we presented the outcome of our Recommenda-

Table 14: Summary of Recommendation Results with various Threshold Values [*BookCrossing Data*]

n_1	n_2	CT	f	No. of Regions	*Recall* (Avg)	*MAE* (Avg)	*RMSE* (Avg)
1000	3000	0.5	0.25	12	0.953	0.7632	0.8544
1000	3000	0.5	0.125	12	0.9175	0.762	0.898
1000	5000	0.5	0.25	8	0.954	0.7775	0.8797
1000	5000	0.5	0.125	8	0.9125	0.7779	0.9363

Table 15: Results of Recommendation [*MovieLens Data*] [$n_1 = 500$; $n_2 = 1000$; $CT = 0.5$; $f = 0.25$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	2844	0.6523	0.8432	0.913	**4500.35**	
1.2	1069	0.5178	0.7321	0.8817	**685.29**	
1.3	1283	0.4934	0.7381	0.8701	**673.45**	5872.44
1.4	319	0.3128	0.402	0.8276	13.35	
1.1.1	248	0.3631	0.5183	0.8296	6.12	
1.1.2	379	0.5764	0.7714	0.8305	18.41	
1.1.3	1884	0.3128	0.4123	0.8601	**1695.5**	**1734.31**
1.1.4	333	0.3253	0.4655	0.8718	14.28	
1.2.2	18	0.2886	0.4456	0.9474	0.12	
1.2.3	1051	0.4779	0.6822	0.898	**619.5**	**619.62**
1.3.1	808	0.4388	0.6208	0.9101	199.5	
1.3.2	475	0.4412	0.6478	0.8701	26.65	**226.15**
1.1.3.1	117	0.4561	0.625	0.8901	0.84	
1.1.3.2	192	0.4529	0.6105	0.8925	2.14	
1.1.3.3	961	0.5001	0.6567	0.9059	230.4	**316.78**
1.1.3.4	614	0.441	0.6297	0.9265	83.4	
1.2.3.1	501	0.4832	0.6536	0.8667	43.2	
1.2.3.2	33	0.4331	0.5964	0.8649	0.4	**106**
1.2.3.4	517	0.4592	0.6116	0.9091	62.4	
	Average	**0.4435**	**0.6147**	**0.8824**		

Table 16: Results of Recommendation [*MovieLens Data*] [$n_1 = 500$; $n_2 = 1000$; $CT = 0.5$; $f = 0.125$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	2844	0.6534	0.8345	0.8936	**4256.5**	
1.2	1069	0.5234	0.7423	0.8817	**537.54**	
1.3	1283	0.4821	0.7524	0.939	**529.4**	5332.89
1.4	319	0.334	0.453	0.8889	9.45	
1.1.1	248	0.3731	0.5234	0.918	4.3	
1.1.2	379	0.5743	0.7823	0.9074	15.3	
1.1.3	1884	0.3012	0.4178	0.8913	**1495.76**	**1527.86**
1.1.4	333	0.3354	0.4765	0.8947	12.5	
1.2.2	18	0.2921	0.4563	1.0	0.11	
1.2.3	1051	0.5032	0.7012	0.9167	**483.5**	**483.61**
1.3.1	808	0.40	0.6178	0.8901	145.63	
1.3.2	475	0.425	0.648	0.8193	20.36	**165.99**
1.1.3.1	117	0.4432	0.631	0.9419	0.75	
1.1.3.2	192	0.4523	0.6278	0.9432	2.56	
1.1.3.3	961	0.511	0.6456	0.9277	216.8	**289.71**
1.1.3.4	614	0.4563	0.7012	0.9545	69.6	
1.2.3.1	501	0.5012	0.6843	0.9155	36.7	
1.2.3.2	33	0.4432	0.6024	0.8421	0.32	**84.45**
1.2.3.4	517	0.4452	0.6223	0.813	47.43	
	Average	**0.4447**	**0.6274**	**0.9041**		

Table 17: Results of Recommendation [*MovieLens Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.25$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	2844	0.6523	0.8432	0.913	4500.35	
1.2	1069	0.5178	0.7321	0.8817	685.29	
1.3	1283	0.4934	0.7381	0.8701	673.45	5872.44
1.4	319	0.3128	0.402	0.8276	13.35	
	Average	**0.4941**	**0.6856**	**0.8731**		

Table 18: Results of Recommendation [*MovieLens Data*] [$n_1 = 1000$; $n_2 = 3000$; $CT = 0.5$; $f = 0.125$]

Region No.	*Total Users*	*MAE*	*RMSE*	*Recall*	*Time (mins)*	*Cum. Time(m)*
1.1	2844	0.6534	0.8345	0.8936	4256.5	
1.2	1069	0.5234	0.7423	0.8817	537.54	
1.3	1283	0.4821	0.7524	0.939	529.4	5332.89
1.4	319	0.334	0.453	0.8889	9.45	
	Average	**0.4982**	**0.6956**	**0.9008**		

Table 19: Summary of Recommendation Results with various Threshold Values [*MovieLens Data*]

n_1	n_2	CT	f	No. of Regions	*Recall* (Avg)	*MAE* (Avg)	*RMSE* (Avg)
500	1000	0.5	0.25	19	0.8824	0.4435	0.6147
500	1000	0.5	0.125	19	0.9041	0.4447	0.6274
1000	3000	0.5	0.25	4	0.8731	0.4941	0.6856
1000	3000	0.5	0.125	4	0.9008	0.4982	0.6956

tion Algorithm in terms of MAE, RMSE and Recall values for the different regions. As for example, in Table 10, region 1.1 has 1594 users and the average MAE, RMSE and Recall values are 0.7613, 0.8379 and 0.9225 respectively averaged over 1594 users. The last row of table 10 shows the values for MAE, RMSE and Recall averaged across all the regions. In Tables 11, 12, 13 we report similar results with different set of threshold values. Analyzing all the Tables, we can conclude that our algorithm is working fine with acceptable values for the different evaluation metrics. In Table 14, we have given a summary of the recommendation results for Book-Crossing Dataset. Here we have reported the Recall, MAE and RMSE values averaged over all the regions. Here Avg is abbreviation for Average. Similarly we present the results of our Recommendation Algorithm performed on the MovieLens Dataset in Tables 15 to 18 for different set of threshold values, and the corresponding summary in Table 19.

We have seen in section 7.2 that Spatial Autocorrelation exists in the decomposed regions and in this section we find that the Recommendation Algorithm applied to those regions separately, significantly improves the overall running time, and at the same time maintain the quality of recommendations (high Recall values). Therefore we can conclude that the idea of recommendation using spatial autocorrelation makes sense.

8. CONCLUSION

In our proposed Spatially Aware Recommender System we have employed a splitting technique to first divide the locations based on the correlation value and then we have mapped each user to a particular location according to the split criteria. Experimental analysis using real datasets shows that our model is efficient and scalable. Further, it provides quality recommendations and also minimizes the computations over irrelevant or less significant data to a large extent without degrading the efficiency of the Recommender System. Online experimentation of the split algorithm and the Recommendation Algorithm will be the focus of our future work. Evaluating the Recommendation Algorithm using other metrics will be a part of our future work. We will also try to use other metrics for finding the spatial correlation and similarities between users with the aim of optimizing the splitting technique and the Recommendation Algorithm.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, pages 734–749, 2005.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer US, 2011.

[3] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, 2011.

[4] B. Bhasker and K. Srikumar. *Recommender Systems in e-Commerce*. McGraw-Hill Education, 2010.

[5] J. Das, S. Majumder, and P. Gupta. Voronoi based location aware collaborative filtering. In *Proceedings of the 3rd IEEE Conference on Emerging Trends and Applications in Computer Science (NCETACS)*, pages 179–183, 2012.

[6] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, 2000.

[7] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 450–461, 2012.

[8] X. Li, Z. Mi, Z. Zhang, and J. Wu. A location-aware recommender system for tourism mobile commerce. In *Proceedings of the 2nd International Conference on Information Science and Engineering (ICISE)*, pages 1709–1711, 2010.

[9] C. P. Lo and A. K. W. Yeung. *Concepts and Techniques of Geographic Information Systems*. Prentice Hall, 2007.

[10] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, 1999.

[11] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrival*, 2011.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e.commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, pages 158–167, 2002.

[13] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[14] E. Zhong, W. Fan, and Q. Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *Proceedings of the SIAM International Conference on Data Mining*, 2012.