Balltree Similarity: A Novel Space Partition Approach for Collaborative Recommender Systems

Hiep Xuan Huynh(✉), Nhung Cam Thi Mai, and Hai Thanh Nguyen

College of Information and Communication Technology,
Can Tho University, Can Tho, Vietnam
hxhiep@ctu.edu.vn

Abstract. The recommender systems have been widely applied in numerous applications that support online retailers, video sharing websites, medical systems, etc. Similar measures are essential in providing valuable recommendations to users in such systems. This work presents a novel approach, namely **Ball-Sim**, with a new similarity metric using a balltree structure for recommender systems. Furthermore, we want to leverage the tree structure to determine the closest k nearby users to improve the recommender systems' efficiency. The work's experimental scenarios outlined the steps of building a balltree and identifying nearby users based on the tree structure. Besides, the work also evaluates the implemented recommender system by comparing the recommender system's results based on the balltree-based spatial partitioning with the recommender system using the default parameters. The data used in the experiments is the Movielens dataset, a web-based film recommender system, and an important data source for evaluating the studies, with 100,000 samples, including ratings from 943 users for 1,664 movies. The results show that the recommender system with a balltree-based similarity metric can improve the accuracy compared to a commonly-used measure such as the cosine metric.

Keywords: Balltree · Similarity measure · Movielens · Recommender systems · Spatial partitioning

1 Introduction

Recommender systems research is a field of machine learning with broad applications in e-commerce, entertainment, and education. The system seeks to predict the "products" for the appropriate "user". In e-commerce, the advisory system helps buyers find suitable goods, helps sellers find potential customers, and boosts sales. The system recommends items according to the user's preferences in entertainment and education. Consulting system opens up research potential to build practical systems to assist users in making decisions. A balltree [19] is a binary tree structure that supports a multidimensional spatial partitioning

C. V. Phan and T. D. Nguyen (Eds.): ICCASA 2022, LNICST 475, pp. 112–128, 2023.
https://doi.org/10.1007/978-3-031-28816-6_9

algorithm, distributing data points to find an efficient separating superspace. In addition, the balltree applies a close neighbor search algorithm to find the required data. Selecting a recommender system model is a problem in building an effective advisory system. The algorithm for finding the nearest user to predict a ranking value for current users' items is also quite complicated. Questions to be addressed: Which recommender system model is used? How do we find the users closest to the user needing advice? How do we evaluate whether the model is being used effectively or not? The work uses a collaborative user-based filtering system with a balltree spatial partitioning algorithm to find the nearest neighbors effectively from the questions raised. The work proposes a new approach: using a balltree structure to store users' lists and then searching for the closest user with the k-neighbor algorithm. Then, after the most recent users are available, use the rating system to calculate the average rating for the current user, suggest items, and finally rate the model.

This study proposes an approach, namely **Ball-Sim**, leveraging the balltree structure and building the balltree-based k nearest neighbors search method to provide a recommender system with a novel similarity measure. We also present detailed examples of using balltree for the recommender system. Our approach performs better than the recommender method with cosine measures on Movielens data, including about 100,000 samples.

2 Related Work

Recommender systems are software and engineering tools that provide product recommendations for users to use [1,2]. In the recommender system, three objects need to be considered: users (users), items (items - collectively called news items, or e-commerce sites such as Amazon, items are recommended products for users). Moreover, user responses to the item (called reviews or ratings). The recommender system is based on many different methods and algorithms but can be divided into three types of approaches [1,3–5]: Content-based Recommender system, Collaborative Filtering Systems, and Knowledge-based recommender systems.

2.1 Content-Based Recommender Systems (CBF)

The systems need to learn to suggest similar items that the user has liked in the past. The emphasis technique analyzes the content (attributes) for prediction. For example, if users often read articles related to Linux operating systems or comment on software engineering-related content, the system will make recommendations based on history and recommend similar content. A content-based advisory system matches one or more characteristics compared to user profiles to determine the relevance of that item to a particular user. Therefore, the advisory process determines which products are most suitable for the user's preferences. A list of product features can be given, such as product name, price, color, etc., and other product-related information that the user can find attractive.

Although the recommender model based on content filtering has been successfully applied in many areas [6–8], there are some disadvantages to this method [6]. The first one may be expertise Focus, where Content-filtering recommender models tend to recommend products similar to what users previously rated. For example, mothers who breastfeed or buy diaper products for their babies regularly review and search for mothers' sites also suggest products about diapers and milk. However, shopping needs may also be related to household gadgets, kitchen utensils, or fashion related by age, but these products are not consulted. The second may be a characteristic citation problem where Content analysis is limited by techniques on the content that require items to be described clearly. This problem will be more difficult when selecting complex content features for multimedia data such as graphics, sound, images, and video. New User Problem [6]: Content filtering recommender models are helpful when users rate or buy a relatively large number of products. However, new users have very few ratings, so the system will not recommend suitable products for users.

2.2 Collaborative Filtering Systems (CF)

Collaborative filtering-based recommender systems recommend products to users based on similar product measurements evaluated by other users. These systems work by building a huge database of information about the user's behavior and preferences (user-item matrix) and predicting what the user will like based on the similarities and comparisons. Collaborative filtering is considered the most common technique in the suggestion system. Collaborative filtering techniques work well when there is enough assessment information. Therefore, the recommender model based on collaborative filtering depends entirely on user rating data for products. For example, in the collaborative filtering recommender model to introduce movies to viewers, the model finds groups of viewers with similar interests as viewers who need previous advice. Then the model introduces movies that this group of viewers appreciates to those who need advice. The collaborative filtering methods [6,7,9] are divided into two main branches: memory-based and model-based. The authors usually use Memory-based approaches to utilize all assessments, products, and users stored in the system, to base these evaluations to render private lists.

2.3 Knowledge-Based Recommender Systems

Knowledge-based recommender systems are helpful in case items are not frequently used, for example, in e-commerce, real estate-related products, and car tourism-related products. However, the user's rating matrix is not informative enough to suggest since most users only buy the product once with different detailed requirements. Therefore, it is not easy for the model to collect enough evaluation information about a product. On the other hand, the user's preferences about the product may change over time. Therefore, the system may combine user ratings, product attributes, and historical knowledge of the similarity between user requirements to suggest matches [10–12].

3 The Proposed Recommendation Method Based on Balltree Structure

3.1 Preliminary on Balltree Algorithm

One data structure to speed the finding of neighbors is the balltree structure. Balltree structure is beneficial in cases where the amount of data is considerable. It can be built for data modeling, supporting adding and deleting data, and processing with multidimensional space. This algorithm is also applied in many fields, such as robots, computer vision, speech processing, and graphics. A balltree is a binary tree with a hierarchical structure. Start with 2 clusters (each is a ball) to be created. Since it is a multidimensional space, each ball is called a matching super-sphere. Some points in a multidimensional space will belong to one cluster, not all. The points will belong to the cluster with the shortest distance. Each ball will include two sub-clusters, and each sub-cluster becomes a ball. This sub-cluster is further divided into the next sub-clusters until a certain depth. Each ball in the balltree contains a node-set of points in Euclidean space. Each balltree node can be either the root node (containing all the original data), the intermediate node (containing the set of points), or the leaf node (one data point). Each node consists of 2 components: center and radius. A node can define parent, left, and right child nodes. If it is a root node, there is no parent. If it is a leaf node, there is no child. Each node in the tree identifies the smallest ball containing all the data points in its subtree.

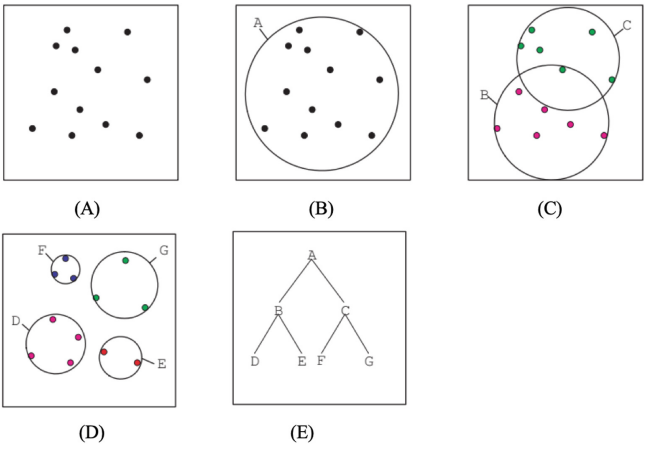

Fig. 1. The steps of building a Balltree.

This study builds the balltree to store user vectors as the recommender system's original data according to a top-down algorithm.

Figure 1 exhibits the steps of building a Balltree where (A) is a set of data, (B) is a root node of the tree, (C) is from the root node divided into two children

nodes, (D) is each child node is divided into the underlying child nodes, and (E) is a balltree with the nodes of D, E, F, G considered as the leaf nodes.

The whole top-down algorithm's complexity when building balltree is $O(n * log^2 n)$. The top-down method of balltree construction is a recursive process that starts at the root node, then the intermediate nodes, and finally, the leaf nodes. For example, calling pi the initial node, dividing pi into two subnodes consists of four main steps [13] (as shown in Algorithm 1).

Algorithm 1: Create a balltreee structure: create-balltree(D)

```
Result: A balltree
if If D only has a point then
    create a leaf node B as a point in D ;
    return B;
else
    Call L, R are two child leaves which will be created from the
     considering node;
    Create B with 2 children with steps as follows:;
    Calculate the radius r of the circle (the distance from farthest point f
     to center c) ;
    Center = c (Determine center c);
    B.radius = r (Determine the farthest point f in the data set from
     center c);
    B.leftchild = create-balltree (L) ;
    B.rightchild =create-balltree (R);
    return B;
end
```

- Step one: choose the point farthest from the center of pi, the point piL (left).
- Step two: choose the point farthest from piL, the point PIR (right).
- Step three: assign the data points closest to piL or piR.
- Step four: divide the node pi into 2 subnodes viL or viL, and calculate the center and radius for these two subnodes. The algorithm ends when viL and viL are a point in the original data

Figure 2 shows the process of clustering data with Balltree. The blue line is the super-flat line that divides the initial node into two children. The above process is recursive until each leaf node contains an original data point. After constructing the tree structure or the construction of the balltree, perform an efficient search of the constructed structure using k-nearest neighbors (kNN) based on the magnitude and distance between the original data.

3.2 Determining the Closest Neighbors in the Balltree

Defining the query object is complicated and time-consuming from a large amount of data. However, among the algorithms built to query data, k-Nearest neighbor (k-NN - nearest neighbor) is quite efficient and straightforward. The

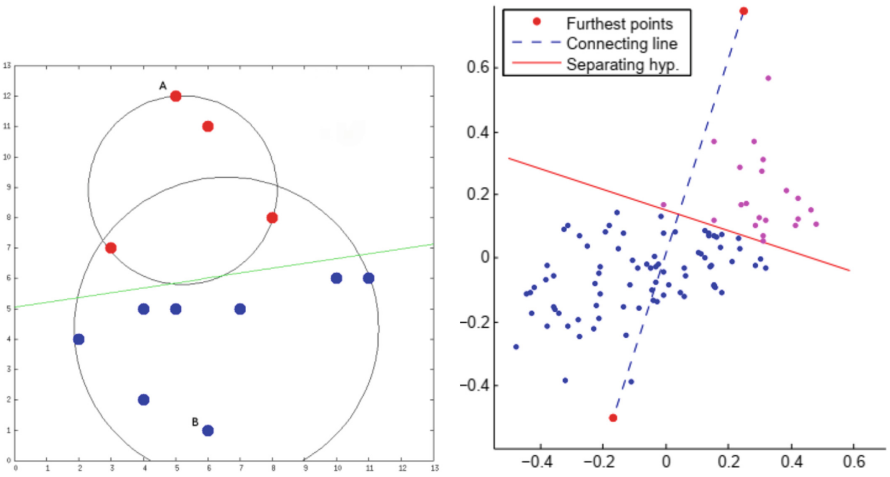

Fig. 2. An illustration of clustering data with Balltree [13].

data structures are applied to k-NN, such as balltree, k-d tree, Principal Axis Tree (PAT), Orthogonal Structure Tree (OST), Nearest Feature Line (NFL), and Center Line (CL). The balltree structure is chosen because it is efficient in multidimensional space. In the multidimensional space, we denote V as a set of data points (user set is the rows considered in the matrix), Q contains the neighbor points of query q in V, and K is the latest number of users to search for a new user. Q contains the k users closest to query q if and only if the maximum distance from the query point q to the points in Q.

The maximum possible distance from the query point q to the points in B is calculated by the formula 1.

$$D = \begin{cases} \infty & if|Q| < k \\ max_{x \in Q}|x - q| & if|Q| \leq k \end{cases} \tag{1}$$

Likewise, the maximum possible distance from the query point q to the points in B is calculated by the following formula 2:

$$D_T = \begin{cases} max(|q - T.centroid| - T.radius, D_{B.parent}) & T \neq root \\ max(|q - T.centroid| - T.radius, 0) & T = root \end{cases} \tag{2}$$

The algorithm starts the search from the root node. During the search process, the algorithm recalculates Q. At each node B, the algorithm performs one of three cases and returns Q containing k positions with the same closest query condition of query q (detailed in Algorithm 2). Case one: if the distance from the query point q to the considered node T is more significant than D, ignore B and return Q. Case two: if T is a leaf node, go through all the points and update

Q. Case three: if T is an inner node (ball, not a leaf node), call recursive search algorithm for B's two children: left and right.

Algorithm 2: The algorithm to search a subtree in a balltree: searchBallSubtree (k, q, Q, node)

```
Result: A balltree
if D_B < D then
    return Q ;
else
    if B is a leaf then
        while x in B.Points() do
            if |x − q| < D then
                add x to Q;
                if |Q| = k + 1 then
                    remove the furthest neighbor from Q;
                    Update D;
                end
            end
        end
    else
        let child1 be the child node closest to q ;
        let child2 be the child node furthest from q ;
        searchBallSubtree (k, q, Q, child1) ;
        searchBallSubtree(k, q, Q, child2) ;
    end
end
```

We consider an example, including the proposed steps to illustrate the Balltree structure's use to determine k-neighbors and similar users. To illustrate the balltree structure, we assume that we have 8 points in the set M: $M = \{X_j\}, j = 1..8$, with each point having coordinates (x, y). We have a matrix M with eight rows and two columns, as shown in Fig. 3.

	F1	F2
X1	1	1
X2	2	3
X3	3	2
X4	13	20
X5	11	16
X6	5	6
X7	5	8
X8	5	11

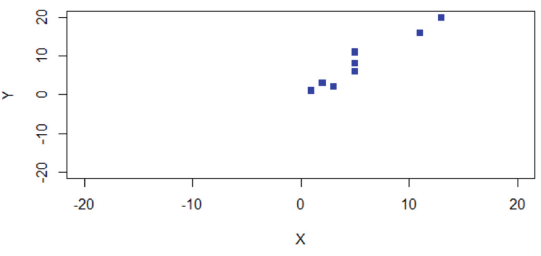

Fig. 3. An example including 8 points, their coordinates, and their visualization.

Building the Balltree from the Given Points. We need to identify the center and radius of nodes from the center in this stage.

Step 1: Determine the center and radius of the parent node of 8 given points:
The center = $(\frac{1+2+3+13+11+5+5+5}{8}, \frac{1+3+2+20+16+6+8+11}{8}) \approx (6{,}8)$. X4 is the farthest point in set A relative to the center (using the formula for calculating the distance between 2 points in the coordinate system). From X4 to the center, we have the radius computed by $\sqrt{(13-6)^2+(20-8)^2} \approx 13.5$.
Step 2: The algorithm expands the root node A to two child nodes, B and C. In set A, X4 is the farthest point from the center of root node A, while X1 is the farthest from X4. We consider X1 and X4 as two points of two sub-circles, B and C, where B=Node(A).child1 = X1, X2, X3, X6, X7, X8 and C = Node(A).child2 = X4, X5. At this time, the center and radius of the circle B are computed by the Center of B is A.child1 = $(\frac{21}{6}, \frac{31}{6}) \approx (3.5{,}5)$, X8 as the farthest point in set B. Center of C = A.child2 = (24/2, 36/2) = (12, 18), determine X4 as the farthest point in set C with the radius C = A.child2=2.23.
Step 3: We continue to divide node B and C into their child leaves of D, E, F, G with C.child1 = F = X4, C.child2 = G = X5, B.child1 = D = X1, X2, X3, and B.child2 = E = X6, X7, X8
Step 4: We continue to divide nodes D and E into their child leaves of H, I, K, and L, respectively. At the node D: the Center of D = ((1 + 2 + 3)/3, (1 + 3 + 2)/3) = (2, 2). Determine the point X2 or X3 is the farthest point in D. The radius of a circle D = 1. At the node E: The center of E = ((5 + 5 + 5)/3, (6 + 8 + 11)/3) ≈ (5, 8.3). Determine the point X8, which is farthest from the center in D, and the Radius of the circle D = 2.7. We have D.child1 = H = X1, D.child2 = I = X2, X3, E.child1 = K = X8 and E.child2 = L = X6, X7

At this step, I and L can be divided into child nodes, but we only consider k=2 nearest neighbors in this example. It should be considered that the algorithm ends. After splitting the nodes, the model shows the binary tree as Fig. 4.

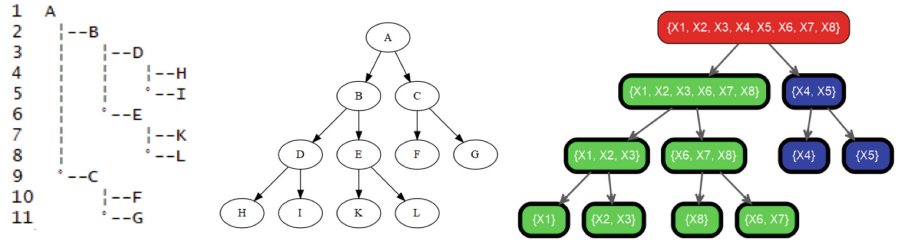

Fig. 4. The Balltree structure illustration to represent the 8 given points.

Determining Neighbours on the Balltree. Suppose that we have a new point, q(7,4). Based on the tree structure constructed in the above steps, we search for the k=2 nearest points to q. The process includes steps as follows.

Step 1: Initialize the values: K = 2, determine the 2 nearest neighbours for q with V = X1, X2, X3, X4, X5, X6, X7, X8 and $Q = \emptyset$ (the set of nearest neighbors of q is initially empty), D_T=0, since at the beginning, T is initially as the root node.
Step 2: T is the root node, so $D_T = 0, Q = \emptyset$. Continue to consider 2 child nodes, B and C.
Step 3: Calculate the distance from q(7, 4) to the center of B (3.5, 5) and C (12, 18), we obtain $D_{qB} = \overline{qB} = D_{qC} = \sqrt{13.25}$, so B is the closest one to q while C is the farthest from q. B is the intermediate node. The recursion can be applied to the left and right child nodes D and E, and update Q = X1, X2, X3, X6, X7, X8.
Step 4: Consider 2 child nodes of B: D = X1, X2, X3 and E = X6, X7, X8. We calculate the distance from q to center D and E: $D_{qD} = \overline{qD} = \sqrt{29}$ and $D_{qE} = \overline{qE} = \sqrt{22.4}$, so E is closest to q, and D is farthest from q. E is the intermediate node, the recursion is applied to E, and update Q = X6, X7, X8.
Step 5: Consider 2 child nodes of E: K = X8 and L = X6, X7 and calculate the distance from q to center K and L. The Center of K: X8 = (5,11). The center of L: ((5 + 5)/2, (6 + 8)/2) = (5,7), and distances between q and K, L is computed by $D_{qK} = \overline{qK} = \sqrt{53}$ and $D_{qL} = \overline{qL} = \sqrt{13}$. From the obtained results, we determine that L is the nearest point to q while K is the farthest from q. L is the intermediate node. We perform the recursion on L's left and right children and update Q = X6, X7. With k = 2 nearest neighbors, the algorithm stops with the return values, including X6 and X7 (Fig. 5).

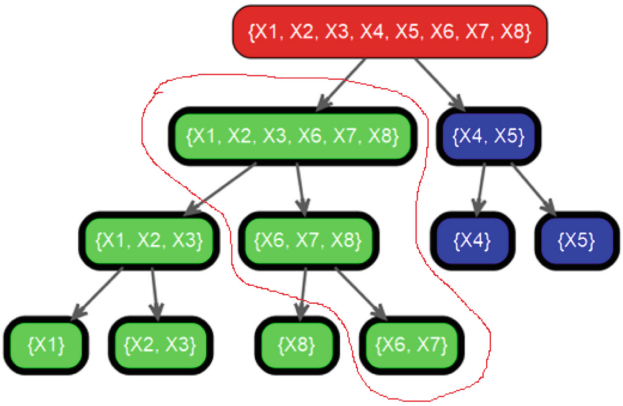

Fig. 5. The red curve covers the branch which is included to exhibit how to determine k = 2 nearest neighbors (X6, X7) of q. (Color figure online)

3.3 Modeling the Problem of Measuring the Similarity of Users in a Recommender System Based on Balltree Structure

Collaborative filtering techniques can be divided into memory-based and model-based approaches. Memory-based techniques include user-based and item-based techniques based on the user's historical or past items' data. This technique consists of three main steps: representing the ranking matrix, Calculating similarity between users or between items, predicting ratings, and making suggestions. We use data mining and machine learning techniques for Model-based techniques to build predictive models based on data collected in the past. These techniques analyze user-item matrices to identify the relationships between items; These relationships are used to compare the list of top-N suggestions. This technique also includes three main steps: Form the ranking matrix, Build models by machine learning method (training phase), and Predict actual data based on the learned model (predictive period). Based on the model, the proposed recommendation method based on balltree spatial partitions is classified into a collaborative filtering group. This study leverages the balltree spatial partition to store the user vectors, calculate their similarity, and predict the result.

Balltree is a complete binary tree data structure built for data modeling, instrumental in multidimensional spatial processing. Balltree is applied in numerous different fields. In this study, the balltree is considered a method to determine the similarity of the user list. Based on a built-in balltree structure, the recommender system determine the closest neighbor to a particular user and ultimately finds advisory values for a new user. Based on the theory of balltree structure, we propose a novel approach in the recommender system with balltree spatial partitions as follows: input data is a matrix, and here is a rating matrix of the user set $U = u_1, u_2, u_3, ..., u_n$ for the set of products - items $I = i_1, i_2, i_3, ..., i_m$. Users are considered a dataset in m-dimensional space, proceeding to build a balltree structure based on that data set. During building a balltree, the two parameters of a node are stored as the node's center and radius (exhibited in Fig. 6). After the balltree has been built, we determine the closest user with the closest k-neighbor algorithm for a given user. At this point, the node's center and radius parameters are used to identify a list of nodes close to the new node using the distance formula.

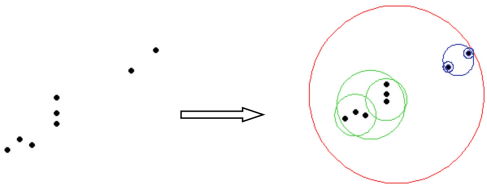

Fig. 6. An illustration of solving the problem with the clusters generated by the balltree approach.

Information Matrix. In a recommender system (RS), the function r is written as follows (Formula 3):

$$r : U \times I \rightarrow R \tag{3}$$

The goal of RS is to find a function $r* : U \times I \rightarrow R*$ such that $\alpha(r, r*)$ satisfies a certain condition. For example, α is a function that estimates the Mean Absolute Error (MAE) (Formula 4). It must be minimal.

$$MAE = \frac{1}{|T|} \sum |r_{i,j} - r*_{i,j}| \tag{4}$$

where $|T|$ is the total rating of the original data set, $r_{i,j}$ is the rating of user u_i for item i_j while $r*_{i,j}$ is the rating predictions of user u_i for item i_j. Creating recommendations using user-based collaborative filtering can be done as shown in Fig. 7.

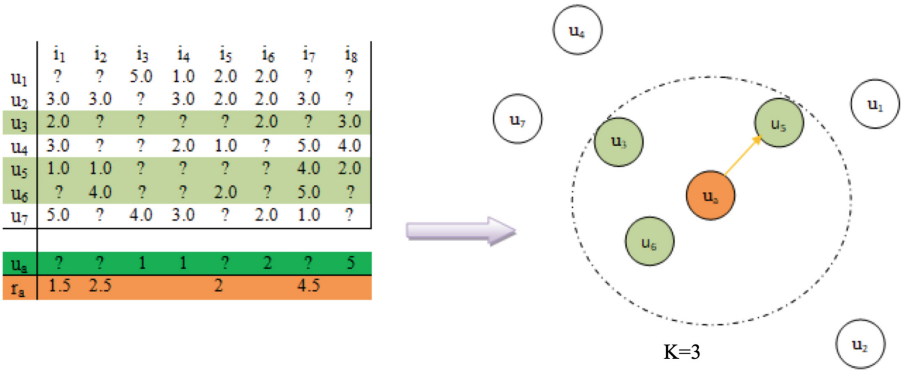

Fig. 7. Suggesting recommendations from the informational function matrix with the k = 3 nearest neighbors.

Providing Recommendations. When a list of users is similar to a user needing to predict, we combine their ratings to generate predictive value. Usually, the predicted results are the average of the values of similar users.

For users who need u_a advice, we determine the results in places that have not yet been validated (question marks are positions that need to be valid). Recommendation results using the predictive function calculate the average value from users similar to u_a.

4 Experimental Results

4.1 Movielens Dataset and Settings for the Experiments

Our method is evaluated on the movielens dataset proposed by grouplens[1]. The dataset includes about 100,000 samples with movies rated by 943 users for 1,664

[1] https://grouplens.org.

movies (with 99,392 samples, including ratings of values ranging from 1 to 5). It is organized in a matrix format of 943 rows, 1,664 columns, and 1,569,152 cells containing users' ratings for movies. However, as we know, not all users watch all movies. In the rating matrix, there are only 99,392 user ratings for the category of movies, including 19 genres of movies (action, adventure, animation, children's, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, scifi, thriller, war, western).

We assume that each node stores one user vector (one row of data in the initial evaluation matrix). After the tree has been built, the leaf nodes' similarity to a new node is calculated as the data to be predicted. The next step is to find k neighbors to find n users closest to the new user. Finally, calculate the advisory results for this new user.

The performances in Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Square Error (MSE) are measured by the average results on 5-fold-cross validation.

To build a recommender system based on Balltree, we use recommendarlab[2] tool package and install additional functions to build a recommender system based on balltree structure: build balltree in recommendarlab package and compute user similarity according to balltree structure combining some functions of sklearn [18] (sklearn. neighbors.BallTree) in python.

Taking examples for scenarios in the following sections, we randomly extract a small 10×10 dataset in the Movielens set after the normalization process (exhibited in Table 1). Then, we divide it into a training set and a test set. For samples used in the examples, we assume that at the i-th repetition of a 5-cross evaluation, there are u_5 and u_6 in the training set, and the remaining is in the test set.

4.2 Scenarios

Three scenarios are presented to show how the proposed method works. In the first one, the system receives input data and defines nodes in the balltree structure. Then, we calculate the similarity of the user vectors by finding neighbors through the balltree tree to determine a list of users similar to new users. The last one identifies the value of reviews and advises new users.

Scenario 1. The system receives the data and defines nodes in the balltree. The considered nodes include root, intermediate, and leaf nodes. Next, we transform the dataset into user vectors where each user is one vector from the original dataset. Finally, the NA values are replaced by the value 0.

[2] https://cran.r-project.org/web/packages/recommenderlab/index.html.

Table 1. Experimental samples for scenarios presented in a matrix of 10×10 where each column represents movies i_n (n = 1..10) and each row represents ratings of the user u_m (m = 1..10) corresponding to the movie.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	5	3	3	4	1	5	2	5	5	5
u_2	4	NA	NA	NA	NA	NA	NA	NA	4	4
u_3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u_4	4	3	NA	NA	NA	NA	NA	NA	NA	NA
u_5	4	NA	NA	2	4	4	NA	4	2	5
u_6	NA	NA	5	5	5	5	3	5	NA	NA
u_7	NA	NA	NA	3	NA	NA	3	NA	NA	NA
u_8	4	NA	4	4	NA	4	4	5	3	NA
u_9	NA	NA	NA	NA	4	5	2	2	NA	NA
u_{10}	NA	NA	5	NA	NA	NA	NA	NA	NA	NA

The training set is illustrated as: $\begin{pmatrix} \overrightarrow{u_1} \\ \overrightarrow{u_2} \\ \overrightarrow{u_3} \\ \overrightarrow{u_4} \\ \overrightarrow{u_7} \\ \overrightarrow{u_8} \\ \overrightarrow{u_9} \\ \overrightarrow{u_{10}} \end{pmatrix} = \begin{pmatrix} 5\,3\,3\,4\,1\,5\,2\,5\,5\,5 \\ 4\,0\,0\,0\,0\,0\,0\,0\,4\,4 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 4\,3\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,3\,0\,0\,3\,0\,0\,0 \\ 4\,0\,4\,4\,0\,4\,4\,5\,3\,0 \\ 0\,0\,0\,0\,4\,5\,2\,2\,0\,0 \\ 0\,0\,5\,0\,0\,0\,0\,0\,0\,0 \end{pmatrix}$

The test set is illustrated as: $\begin{pmatrix} \overrightarrow{u_5} \\ \overrightarrow{u_6} \end{pmatrix} = \begin{pmatrix} 4\,0\,0\,2\,4\,4\,0\,4\,2\,5 \\ 0\,0\,5\,5\,5\,5\,3\,5\,0\,0 \end{pmatrix}$

Step 1: Determine the center and radius of the parent node. From the training set with $A = u_1, u_2, u_3, u_4, u_7, u_8, u_9, u_10$, we determine the center by:

$(2.125\ 0.75\ 1.5\ 1.375\ 0.625\ 1.75\ 1.375\ 1.5\ 1.5\ 1.125)$

Determine the farthest point in set A to the center from the result obtained above, and using the formula to calculate the distance between n points in the n-dimensional coordinate system, we obtain u_1.

Step 2: We have root node A as the first step. We divide the root node A into two child nodes of B and C. In set A, u_1 is the farthest node from the center of root node A. We obtain u_3, which is the farthest one from u_1. We consider u_3 and u_1 as the center of two sub-circles, B and C where B = Node(A).child1 = u_2,u_3,u_4,u_7,u_9,u_{10} and C = Node(A).child2 = u_1,u_8. At this time, the center of child node B is identified as follows. The center of B = A.child1 is computed with the result:

$(1.34\ 0.5\ 0.83\ 0.5\ 0.67\ 0.83\ 0.83\ 0.34\ 0.67\ 0.67)$

u_9 is found as the farthest point in set B. The same way is applied to B as C. However, since C only has two children and suppose to determine two neighbors,

there is no need to decompose C. In the balltree structure, we draw the root node A and its two child nodes of B and C.
Step 3: We continue to divide the node B into the child nodes where $B.child1 = D = u_2, u_3, u_4$ (u_2 is the farthest point from the center of D) and $B.child2 = E = u_7, u_9, u_{10}$ (u_9 is the point farthest from the center of E).
Step 4: We continue to divide node D and E into the child nodes where $D.child1 = F = u_2$, $D.child2 = G = u_3, u_4$, $E.child1 = H = u_9$, $E.child2 = I = u_7, u_{10}$. The balltree is structured after 4 steps, as exhibited in Fig. 8.

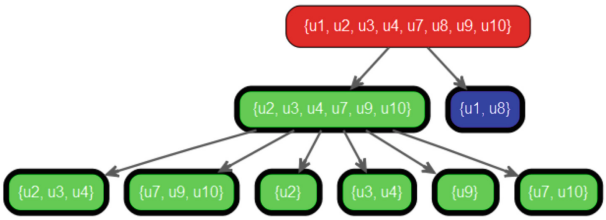

Fig. 8. The structured balltree as an example.

Scenario 2. In this scenario, we calculate the similarity of the user vectors and display a list of the most similar users for the given user.

Taking u_5 as an example, with $u_5 = (4\ 0\ 0\ 2\ 4\ 4\ 0\ 4\ 25)$, the list of users who has the similarity to u_5 as determined as follows.

Step 1: Initialize the values: K = 2, determine 2 neighbors which is the closest to u5 with initialized values of $V = A = u_1, u_2, u_3, u_4, u_7, u_8, u_9, u_{10}$, $Q = \emptyset$, the set of neighbors closest to u5, and $D_T = 0$, since T is initially the root node.
Step 2: Consider T is the root node, so we have $D_T = 0, Q = \emptyset$. Continue with the two child nodes of B and C.
Step 3: To calculate the distance from u_5 to the center of $B = u_2, u_3, u_4, u_7, u_9, u_{10}$ and $C = u_1, u_8$, we compute the distance from u_5 to center of 2 child nodes B (u3) and C(u1) where $D_{u(5)B} = (u(5)B) = 9.85$ and $D_{u(5)C} = (u(5)C) = 6.8$. From the obtained results, we see that C = u_1, u_8 is closest to u5, B = u_2, u_3, u_4, u_7, u_9, u_{10} is the farthest node from u5. Since C has only 2 children u_1, u_8 and we set the neighbors $K = 2$, the algorithm stops. The needed list includes u_1 and u_8.

Scenario 3. In this scenario, we calculate the usage of the recommendation and provide suggestions to the new users. We suggest items for the new users from the pre-stored list of user vectors.

Provide ratings for items for the u_5 based on its 2 closest neighbors, u_1 and u_8. The results obtained from scenario 2 for this Scenario include u_1 and u_8. $u_1 = (5\ 3\ 3\ 4\ 1\ 5\ 2\ 5\ 55)$, $u_8 = (4\ 0\ 4\ 4\ 0\ 4\ 4\ 5\ 30)$. u_5 is determined by calculating the mean of u_1 and u_3, the values of 0 are not included in the evaluation formula $u_5 = (4.5\ 3\ 3.5\ 4\ 1\ 4.5\ 3\ 5\ 4\ 5)$.

4.3 The Experimental Results on 100,000 Samples of Movielens Dataset

The work processed the Movielens matrix before being put into the recommender model. First, data are filtering advancement: each user watches at least 50 movies, and each movie has at least 100 users viewing and rating. After filtering data, the matrix result consists of 560 rows and 332 columns. Data representation using a Heat chart in R shows the user's rating in the Movielens set after filtering the data. Proceed to divide into the training data set and test data set. If dividing the data by 80% division is training set, and 20% is test set, the training data set results include 463 users ($463 \times 332 = 45248$ reviews) and a test set of 97 people used ($97 \times 332 = 10050$ ratings).

The balltree-based recommender results of the model are exhibited in a matrix format with a structure of 10×111 (each column is one user, and each cell is a movie selected to present to the user in the corresponding column). For example, Fig. 9 shows consultation results for the first four users, with each user choosing the ten highest-ranked movies.

```
     2
 [1,] "Blade Runner (1982)"
 [2,] "Silence of the Lambs, The (1991)"
 [3,] "Schindler's List (1993)"
 [4,] "Casablanca (1942)"
 [5,] "Raiders of the Lost Ark (1981)"
 [6,] "Monty Python and the Holy Grail (1974)"
 [7,] "Big Night (1996)"
 [8,] "Boot, Das (1981)"
 [9,] "Amadeus (1984)"
[10,] "Twelve Monkeys (1995)"

     38
 [1,] "Star Wars (1977)"
 [2,] "Fargo (1996)"
 [3,] "Good Will Hunting (1997)"
 [4,] "Silence of the Lambs, The (1991)"
 [5,] "Shawshank Redemption, The (1994)"
 [6,] "Postino, Il (1994)"
 [7,] "Leaving Las Vegas (1995)"
 [8,] "Secrets & Lies (1996)"
 [9,] "Boot, Das (1981)"
[10,] "Usual Suspects, The (1995)"

     11
 [1,] "Star Wars (1977)"
 [2,] "Godfather, The (1972)"
 [3,] "L.A. Confidential (1997)"
 [4,] "Secrets & Lies (1996)"
 [5,] "Leaving Las Vegas (1995)"
 [6,] "Titanic (1997)"
 [7,] "Trainspotting (1996)"
 [8,] "Good Will Hunting (1997)"
 [9,] "Raiders of the Lost Ark (1981)"
[10,] "Apt Pupil (1998)"

     49
 [1,] "Godfather, The (1972)"
 [2,] "Graduate, The (1967)"
 [3,] "Big Night (1996)"
 [4,] "Leaving Las Vegas (1995)"
 [5,] "Alien (1979)"
 [6,] "Boot, Das (1981)"
 [7,] "Taxi Driver (1976)"
 [8,] "Dr. Strangelove or: How I Learned …
 [9,] "Rear Window (1954)"
[10,] "Secrets & Lies (1996)"
```

Fig. 9. An illustration of suggesting 10 movies for the first 4 users.

We compare the accuracy of the proposed model with the balltree to the cosine measure (the similarity between 2 users of u and v is computed by Formula 5), a widely-used method in recommender systems [15–17]. The results in Table 2 show that applying a tree structure to a collaborative human-based filter model can improve the model's accuracy.

$$sim(u, v) = cos(\overrightarrow{v}, \overrightarrow{u}) = \frac{\overrightarrow{v} * \overrightarrow{u}}{||\overrightarrow{v}|| * ||\overrightarrow{u}||} \tag{5}$$

Table 2. Performance comparison in average RMSE, MSE and MAE on 5-fold-cross validation between two approaches

	RMSE	MSE	MAE
UBCF_Balltree	0.9625551	0.9774664	0.7708005
UBCF_Cosine	0.9693691	0.9863076	0.7727980

The figure below shows results comparing the two models' accuracy. The results show that RMSE, MSE, and MAE of User-based collaborative filtering with the Balltree (UBCF_BALLTREE) model are better than these indicators on the User-based collaborative filtering method using Cosine Similarity (UBCF_Cosine).

5 Conclusion

The work focuses on proposing a novel measure that can improve the recommender system's efficiency by combining the recommender system under the user collaborative filtering model based on the balltree binary tree structure. Experiments with **Ball-Sim** on Movielens dataset have achieved a promising performance compared to a commonly-used similarity measurement as cosine. The work also introduced numerous scenarios with detailed steps.

Besides the achieved results, the study's next development direction is to build a recommender system based on numerous different datasets to compare results. Also, there are many algorithms for building balltree trees; the article is expected to begin further studies on balltree-based recommender systems with more diverse algorithms.

Acknowledgment. We want to express our great appreciation to Dr. Nghia Trung Duong, Can Tho University of Technology, and Dr. Lan Phuong Phan, Can Tho University, for their valuable and constructive suggestions during the planning development of this research work.

References

1. Aggarwal, C.C.: Knowledge-based recommender system. In: Recommender Systems, pp. 15–19. Springer, Heidelberg (2016)
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 191–226. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_6
3. Felfernig, A., Jeran, M., Ninaus, G., Reinfrank, F., Reiterer, S., Stettinger, M.: Basic approaches in recommender systems. In: Robillard, M., Maalej, W., Walker, R., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering, pp. 15–37. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45135-5_2
4. Bauman, K., Tuzhilin, A.: Location-based recommender systems. In: Encyclopedia of GIS, pp. 43–92. Springer, Heidelberg (2017)

5. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems, Foundations and Trends in Human-Computer Interaction, SIR Ranking of United States, pp. 1–94 (2011)
6. Aggarwal, C.: Recommender Systems: The Textbook. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29659-31
7. Isinkaye, F.O., Folajimi, Y.O., Ojokoh, B.A.: Recommender systems: principles, methods, and evaluation. Egypt. Inform. J. **16**, 261–273 (2015)
8. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005)
9. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004). ISSN 1046–8188
10. Felfernig, A., Teppan, E., Gula, B.: Knowledge-based recommender technologies for marketing and sales. Int. J. Pattern Recognit. Artif. Intell. **21**(02), 333–354 (2007)
11. Burke, R.: Knowledge-based recommender systems. Encycl. Libr. Inf. Syst. **69**, 175–186 (2000)
12. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
13. Dolatshah, M., Hadian, A., Minaei-Bidgoli, B.: Ball*-tree: efficient spatial indexing for constrained nearest-neighbor search in metric spaces. Iran University of Science and Technology (2015)
14. Hua, J., Lianga, J., Kuang, Y., Honavar, V.: A user similarity-based top-N recommendation approach for mobile in-application advertising. School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (2018)
15. Singh, R.H., Maurya, S., Tripathi, T., Narula, T., Srivastav, G.: Movie recommendation system using cosine similarity and KNN. Int. J. Eng. Adv. Technol. (IJEAT) **9**(5), 556–559 (2020). ISSN 2249-8958
16. Gupta, M., Thakkar, A., Aashish, Gupta, V., Rathore, D.P.S.: Movie recommender system using collaborative filtering. In: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, pp. 415–420 (2020). https://doi.org/10.1109/ICESC48915.2020.9155879
17. Periyasamy, K., Jaiganesh, J., Ponnambalam, K., Rajasekar, J., Arputharaj, K.: Soft cosine gradient and gaussian mixture joint probability recommender system for online social networks. Analysis and performance evaluation of cosine neighbourhood recommender system. Int. Arab J. Inf. Technol. (IAJIT) **14**(5), 747–754 (2017)
18. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. JMLR **12**, 2825–2830 (2011)
19. Omohundro, S.M.: Five balltree construction algorithms. ICSI Technical Report TR-89-063 (1989)