**Jaccard with Singular Value Decomposition Hybrid Recommendation Algorithm**

Chen XU¹, Ling CHEN¹,² and Ben-jie FAN²

¹Department of Computer Science, Yangzhou University, Yangzhou, 225009, China
²State Key Lab of Novel Software Tech, Nanjing University, Nanjing, 210093, China

**Keyword:** Jaccard, Singular value decomposition, Hybrid recommendation.

**Abstract.** In this paper, a comprehensive recommendation algorithm is studied, and a combination of Jaccard and singular value decomposition algorithm is proposed to improve the recommendation accuracy and recall rate. First, we use the Jaccard algorithm to find their recommended listings ranking matrix, which joined the incremental update. Then we use SVD algorithm to complete Jaccard matrix ranking algorithm, so as to obtain a complete list of recommendations, which are recommended according to actual requirements. Experimental results show that the proposed algorithm has higher performance compared with other related recommendation algorithms.

**Introduction**

With the rapid development of Internet technology, the amount of information in the network has brought a sharp rise, the problem of information overload, namely: the amount of information presented so that users can not get the interesting and useful information on their own, so the using efficiency of information and reduce. On the other hand, a large number of information that no one shows any interest became the dark information in the network. Personalized recommendation system came into being, which is a very effective way to solve this problem. It is based on the user's information needs, interest, etc., will be the user interested in the product, the information is recommended to the user. Compared with the search engine, it is recommended to make personalized computing by studying the user's interest preference. The system can find the user's interest point, so as to guide the users to find their own information needs. A good recommendation system can not only provide users with personalized service, but also to establish a close relationship with the user, so that users can rely on the recommendation. Recommendation algorithm is the core of the personalized recommendation system. The idea of this paper is to mix the second kinds of recommendation algorithms in the mixed recommendation: in the framework of some kind of recommendation strategy, and mix the other recommendation strategies, namely: Jaccard with SVD.

**Related Work**

**Jaccard**

The basic idea of Jaccard index: we define the number of neighbors of the X node as \( \Gamma(X) \) in the internet. The similarity between two nodes can be defined as the number of X and Y and common neighbor nodes by dividing the number of node X and Y elements in the Union, the formula is as follows:

\[
Jaccard(X, Y) = \frac{\Gamma(X) \cap \Gamma(Y)}{\Gamma(X) \cup \Gamma(Y)}
\]

First, we carry out the operation of the molecule and retain the elements that the molecule is not 0. Then, we compute the denominator to get the number of the two node element set by the sum of the nodes degree subtracting the intersection of the two nodes. In experiment, we use the Jaccard algorithm to calculate the similarity matrix between houses and find that if the common click
between the two listings is 0, the value of the molecular term is 0. In order to solve this problem, we introduce a method of singular value decomposition (SVD).

**Singular Value Decomposition (SVD)**

The basic idea of singular value decomposition (SVD)\(^{[15-18]}\): We assume that the user - Housing rating matrix \( R \) is \( M \times N \) matrix, and make the matrix \( R \) divided into three parts \( S, U, V \). \( U \) is the \( M \times M \) matrix. \( V \) is the \( N \times N \) matrix. \( S \) is the \( N \times M \) matrix, and the non-diagonal elements of \( S \) are 0. The diagonal elements are singular values of matrix \( R \), and are satisfied from the order of large to small order. Its formula is expressed as follows: \( R = U \times S \times V \).

Singular value decomposition has an advantage, which allows the existence of a simplified approximation matrix. For the matrix \( S \) to retain the \( K \) singular value, the other with 0 to replace. In this way, we can simplify the dimension of \( S \) to a matrix (\( K<N \)) with only \( K \) singular values. Because of the introduction of 0, so you can delete the \( S \) median of 0 rows and columns, get a new diagonal matrix \( S_K \). If the matrix \( U \) and \( V \) are also in accordance with this method, we can get the \( K \times N \) matrix \( U_K \) and \( K \times M \) matrix \( V_K \), then the reconstructed matrix \( R_1 = U_K \ast S_K \ast V_K \). The \( R_1 \) approximation is equal to \( R \), so that the dimension reduction of the matrix is achieved. Schematic diagram is as follows:

![Figure 1. Singular value decomposition.](image)

We want to do is on the recommendation of the housing, so we need to take out the \( V_K \) matrix and calculate the similarity of the matrix by the Euclidean distance.

The basic idea of Euclidean distance: Euclidean metric (also known as Euclidean distance) is a commonly used distance definition, refers to the actual distance in \( m \) dimensional space between two points, or vector natural length (i.e., the distance to the origin). Euclidean distance in two dimensional and three dimensional space is the actual distance between two points. We consider each line of the matrix \( V_K \) as a \( N \) dimensional vector, and then compute the Euclidean distance between rows and rows to obtain the similarity matrix.

**Algorithm Framework and Process**

First, we use Jaccard algorithm to calculate the recommended index ranking matrix; then we use SVD algorithm to complete Jaccard matrix ranking algorithm Index, so as to obtain a complete list of recommended ones. Schematic as shown below:

![Figure 2. Algorithm framework and process.](image)
Incremental Algorithm Design

The Basic Idea of the Algorithm

Jaccard update mechanism: Offline Jaccard algorithm complexity is \( O(M^*N^2) \), where \( M \) for the number of users, \( N \) for the number of houses. We designed an incremental Jaccard algorithm to reduce the time complexity. Drawing as follows:

![Figure 3. Incremental flow chart.](image)

Incremental log extracted 0-1 score array: the number of users \( M' \), the number of listings is still \( N \). We will be five minutes of new data is divided into new users and old users, old users that: before there is a user to click on the user. We analyze the incremental log score matrix:

- Case 1 new users click on a suite, did not change the intersection of the first row of the table and set in the table value, the value of Jaccard.
- Case 2 new users click on two or more units of housing: any two suites \((i, j)\), on the intersection of the following table and set the table: \( \text{int}_{i,j} = \text{int}_{i,j} + \); \( \text{uni}_{i,j} = \text{uni}_{i,j} + \).
- Case 3 The old users click on a suite, and the suite has been visited: no impact on the Jaccard value.
- Case 4 The old users click on a suite, and the suite has not been visited: the user is located in the original 0-1 scoring array of the first line, and scan all non housing. If \( u_{-k} = 0 \), else \( \text{int}_{i,k} = \text{int}_{i,k} + \); \( \text{uni}_{i,k} = \text{uni}_{i,k} + \). If \( u_{-k} = 1 \), else \( \text{int}_{i,k} = \text{int}_{i,k} \); \( \text{uni}_{i,k} = \text{uni}_{i,k} \).
- Case 5 The old users click on a number of suites, and these rooms have been visited before: no impact on the Jaccard value.
- Case 6 The old user clicks more suites, and these rooms have not been accessed: these rooms update intersection table and Set table by case 4.
- Case 7 The old user clicks more suites, and Some of the rooms have been clicked: We filter out the houses that have not been visited and update intersection table and Set table by case 4.

Algorithm Time Complexity Analysis

Because the intersection and union will form a calculation table in every day, so you do not need to add complexity analysis.

- case 1: 0; case 2: \( O(M^*N^2) \); case 3: 0; case 4: \( O(M^*N) \); case 5: 0; case 6: \( O(M^*N^2) \); case 7: \( O(M^*N^2) \).

In summary, the worst time complexity of the incremental Jaccard method is \( O(M^*N2) \). If the time interval is shorter, the user click to the different houses to \( N' \) in this incremental period of time...
M'. This means that the incremental calculation of Jaccard can save a lot of time. According to previous experience, if 5 weeks of log is used as a training set, $M' = 1300$, 5 min average incremental user volume is MAX $M' = 40$. According to the average click on 5 sets of different listings, the efficiency of at least $13000/40 \approx 1 \times e^2$ times.

We made five minutes of new data be divided into new users and old users, old users that: before there is a user to click on the user. For the old users, we will change the original matrix of the corresponding data; for the new users, we add the corresponding line number in the original matrix, calculate the similarity, and storage data. We ignore the old user's data, and calculate the new data that is all as new user data. 5 minute data stream contains new user clicks, but also the old user clicks. We take 5 minutes of all the data as a new user's click behavior. We analyze the accuracy and the error of the algorithm under the Jaccard update mechanism:

![Figure 5. Error analysis and comparison.](image)

In the figure, the red line indicates that the Jaccard algorithm to click on the accuracy rate. The blue line indicates that the incremental Jaccard algorithm to click on the accuracy rate, we can see that the error between the two is very small. In the Jaccard update, we can take 5 minutes of data flow as a result of the new user clicks.

**Experimental Data Selection and Algorithm Evaluation Index**

**Data Set**

We took a sample of a company's mobile phone for three months as an experimental data set, as shown in the following table:

<table>
<thead>
<tr>
<th>Topological properties of data sets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
</tr>
<tr>
<td>Three month</td>
</tr>
</tbody>
</table>

The analysis of Table 1: A total of 1783 houses, a total of 24803 users, the sparsity of only 0.0061

**Data Analysis**

We selected a company three months of data as the experimental data set. The data analysis is carried out on the data set, count the number and click the new day within three months of the new user, and click new is the number of listings. Shown in figure 6:

From Figure 6, three months and 90 days of data. With the passage of time, the number of new hits per day, the number of new users per day, the number of rooms per day is clicked, the total number of users are increasing.
Evaluation Index

**Ranking Score.** Score Ranking focuses on the location of the edge of the test set in the final order. \( H = U - ET \) is set for unknown edges. \( R_i \) represents the unknown edge \( i \in EP \) ranking in the ranking. The unknown edge of the Score Ranking value is \( RS_i = \frac{r_i}{|H|} \), \(|H|\) represents the number of elements in a collection \( H \). Traversing all the edges in the test set, the Score Ranking value of the system is:

\[
RS = \frac{1}{|E^p|} \sum_{i \in E^p} RS_i = \frac{1}{|E^p|} \sum_{i \in E^p} \frac{r_i}{|H|}
\]

Obviously, the smaller the value of \( RS \), the higher the accuracy of the algorithm.

**Precision.** Precision only considers the target user score list in \( L \) before the boundary is accurate prediction. The Precision is defined as:

\[
\text{precision} = \frac{m}{L}
\]

Obviously, the higher the Precision value, the higher the accuracy of the algorithm.

**Hits.** Hits refers to the user to click on the recommended items accounted for the proportion of total users click on the project. Suppose we recommend 10 suites, and then the user to click on a. If the user clicks on the room in the recommended 10 suites, it is recorded as a successful hit. We set \( M \) to represent the total number of users to click on the house, \( L \) said the total number of houses were recommended to be. Hits value is:

\[
\text{Hits} = \frac{L}{M}
\]

Obviously, the higher the value of Hits, the higher the rate of recommendation algorithm.

**Comparison and Analysis of Algorithm Performance**

We performed off-line testing of different algorithms on the dataset, and the results are as follows:
Table 2. Experimental Result.

<table>
<thead>
<tr>
<th></th>
<th>RS</th>
<th>Precision</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>0.5082</td>
<td>5.3750e-004</td>
<td>37.1497%</td>
</tr>
<tr>
<td>Salton</td>
<td>0.4737</td>
<td>0.0025</td>
<td>40.8557%</td>
</tr>
<tr>
<td>Jaccard</td>
<td>0.4353</td>
<td>0.0311</td>
<td>41.1871%</td>
</tr>
<tr>
<td>Sorenson</td>
<td>0.4354</td>
<td>0.0300</td>
<td>41.1871%</td>
</tr>
<tr>
<td>HPI</td>
<td>0.7368</td>
<td>2.0767e-004</td>
<td>3.7662%</td>
</tr>
<tr>
<td>HDI</td>
<td>0.4350</td>
<td>0.0311</td>
<td>40.0723%</td>
</tr>
<tr>
<td>LHN</td>
<td>0.4642</td>
<td>0.0026</td>
<td>1.2052%</td>
</tr>
<tr>
<td>AA</td>
<td>0.5046</td>
<td>0.0014</td>
<td>38.0838%</td>
</tr>
<tr>
<td>RA</td>
<td>0.5014</td>
<td>0.0032</td>
<td>39.3793%</td>
</tr>
<tr>
<td>PA</td>
<td>0.4370</td>
<td>0.0266</td>
<td>13.8295%</td>
</tr>
<tr>
<td>NMF</td>
<td>0.4405</td>
<td>0.0227</td>
<td>28.834%</td>
</tr>
<tr>
<td>Jaccard With SVD</td>
<td>0.4753</td>
<td><strong>0.0311</strong></td>
<td><strong>42.1671%</strong></td>
</tr>
</tbody>
</table>

From the table data can be found, we compare the three evaluation indicators found that the performance of the Jaccard With SVD algorithm is optimal. So we designed the method of combining Jaccard and SVD is reasonable.

Conclusions

In this paper, we study the hybrid recommendation algorithm of Jaccard and SVD. The experimental results show that compared with the previous algorithms, such as: NMF, AA, RA and so on; Jaccard With SVD algorithm has a higher hit rate, better accuracy, better accuracy. At the same time, The increment that we add can make users to more quickly get their recommend the latest results witch their want.

References


