Hindawi
Security and Communication Networks
Volume 2023, Article ID 9869426, 1 page
https://doi.org/10.1155/2023/9869426

Retraction

Retracted: Building an Urban Smart Community System Based on Association Rule Algorithms

Security and Communication Networks

Received 11 July 2023; Accepted 11 July 2023; Published 12 July 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Z. Wenwen, "Building an Urban Smart Community System Based on Association Rule Algorithms," *Security and Communication Networks*, vol. 2022, Article ID 8773259, 11 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 8773259, 11 pages
https://doi.org/10.1155/2022/8773259

Research Article

Building an Urban Smart Community System Based on Association Rule Algorithms

Zhou Wenwen

Shanxi Vocational University of Engineering Science and Technology, Jinzhong 030600, Shanxi, China

Correspondence should be addressed to Zhou Wenwen; zhouwenwen@sxgkd.edu.cn

Received 7 May 2022; Revised 20 June 2022; Accepted 5 July 2022; Published 19 July 2022

Academic Editor: Mukesh Soni

Intelligent system development is an integral component of smart community development and has a significant impact on the development of smart communities. Some cities continue to implement personalized smart community services, resulting in the formation of smart city communities with unique characteristics. Urban smart communities are based on the principle of owner-occupant convenience, integrating a wealth of community information and making it more relevant to each and every resident through intelligent management. Increasing information transmission rates have enhanced the ability of smart community systems to integrate information, but the smart community recommendation method is still based on traditional categorized recommendations. This paper addresses the deficiency of recommended information in smart urban communities. By analyzing user interaction and operation data, we can determine the interest and recognition of browsing attractions among users. Compared to conventional classification recommendations, weighted association rules can identify potentially very important rules applicable to small groups, thereby meeting the needs of various groups and enabling personalized services. Through continuous feedback from user behavior data, the system gradually identifies the community information that users are interested in during the specific recommendation process. After testing, the smart community system's recommendation accuracy and real-time performance have vastly improved in comparison to categorical recommendations, and it can effectively meet the needs of tenants for community recommendations.

1. Introduction

The urban smart community is a collection of diverse human, physical, inanimate, and organizational resources. Its ascent ensures that computer applications will expand into the field. The urban smart community system is an extension of the traditional tourism approach, a platform for intelligent community services centered on personalized user experience and advanced computer technologies, such as the Internet of Things, cloud computing, artificial intelligence, big data, and next-generation communication technologies [1]. The intelligent community system is a vast service system that provides users with great convenience and increases the value of businesses.

As a vast service system, the urban smart community reflects the state-of-the-art in technology, software, and hardware. For instance, in computer technology, cloud computing, artificial intelligence, and other theoretical methods are used, and in software design, new software development tools and software design concepts are utilized. Hardware-wise, portable terminal devices are used to interact with users, and various information resources are actively provided to residents so that they can locate the community resources in which they are interested more quickly and arrange and adjust their work and life plans to achieve a more convenient, comfortable, and liberated living experience [2]. These three features through the monitoring of community residents' life opinions and data analysis, the intelligent community system can unearth the hotspots of life and residents' interests, as well as direct the community services and enterprise products to serve the community users in a targeted fashion.

In this paper, we employ a multiple minimum support weighted association rule mining algorithm to analyze the online behavior of users and discover the association between users' browsing community services in order to

provide a foundation for community service recommendation. Our community service recommendation algorithm offered the following benefits: First, the algorithm can take into account the needs of different individuals and provide personalized services for different users. Second, the rules can be updated over time, and with the passage of time, the algorithm can become more sophisticated. Second, the rules can be updated in a timely manner because the interest level of users in community services will change over time, and the rules are updated in a timely manner to ensure that the entire system remains consistent with the current general interest level of users.

2. Related Work

2.1. Classification of Association Rules. With the continuous research on association rule mining, the mining forms and related mining algorithms have become very diverse. Different criteria have different classifications, which are classified as follows [3].

2.1.1. Classification According to the Type of Processed Values. Numerical and Boolean association rules—Numerical association rules deal with numerical data, which require partitioning the data into different intervals and treating an interval as an attribute. In addition, this type of rule can also have kind variables and can treat a kind as an attribute, and Boolean association rules are for discrete kinds of data, which represent the relationship of whether items with relevance exist or not. For example: Gender = "Male" => Occupation = "Senior Engineer" => Age = "30–35," where gender and occupation are Boolean association rules, and age is a Boolean association rule involving gender and occupation.

2.1.2. Classification by Data Abstraction Level. Single-level association rules do not take into account the different levels of data in real life and ignore the properties of different abstraction levels. For example, buy product = "Tresor milk" => buy product = "puff," which is a single-level association rule of the same concept level. The multilayer association rule makes up for this aspect of the single-layer association rule by taking into account the multilayer nature of the data in real life. For example, buy product = "Ultrabook" => buy product = "Printer," which is a multilayer association rule between detail level and higher level.

2.1.3. Classification According to the Number of Data Dimensions. Unidimensional and multidimensional association rules—Unidimensional association rules involve only one dimension of the processed data and reveal the relationship between the unidimensional attributes of the processed data, such as the goods purchased by the consumer. Multidimensional association rules involve two or more dimensions of the processed data and reveal some relationships between multiple attributes of the processed data, such as the classic story of "beer and diapers."

2.2. Association Rule Mining Objectives and Mining Process. The goal of association rule mining is to find a certain correlation or law, that is, an association rule, hidden among some data attributes and provide a practice guide. In a nutshell, association rule mining is the process of refining data into information, condensing information into knowledge, and then applying knowledge to practice [4]. The ultimate goal is to extract knowledge from data so that practice can be guided. For example, by analyzing consumer shopping habits, we can learn about their purchasing patterns and use that information to guide supermarkets in shelf design and product promotion; by analyzing drug sales in a specific area of pharmacies, we can predict infectious diseases and prevent them in advance; by analyzing stock quotes, we can forecast the stock market situation; and by analyzing data about users' friends' relationships, we can recommend friends to users to help them make better decisions.

The process of association rule mining:

(1) Find all frequent sets L in dataset D, that is, the set of items whose support is greater than or equal to supmin.

(2) Find the strong association rules from the frequent set, that is, the association rules whose confidence is greater than or equal to conmin from the results obtained in (1)

Step (1) is the core of association rule mining, while step (2) is relatively simple. To get the result of step (1), the database must be scanned several times, thus a lot of time will be consumed in I/O operations, which is a key step affecting the efficiency of the whole algorithm.

2.3. Apriori Algorithm. The Apriori [5] algorithm employs an iterative method of layer-by-layer searching, which is simple and straightforward to comprehend, but is only applicable to small data sets. Moreover, the database must be scanned multiple times during its execution, with each scan yielding a candidate set until the longest candidate set is obtained. Since the number of times the Apriori algorithm scans the database is determined by the length of the final longest frequent set, it has high I/O overhead, high computational effort, and high time cost, and the cost of this algorithm will increase in a geometric fashion as the amount of database data increases.

2.3.1. Apriori Algorithm Improvement. Based on the Apriori algorithm, many scholars have proposed improved algorithms to reduce the I/O overhead or avoid the problem of candidate set inflation [6]. For example, AprioriTid algorithm is to calculate the support of frequent sets at the same time when scanning the database for the first time, and then use the candidate sets obtained from the first scan to combine. AprioriHybird algorithm is the product of

combining Apriori algorithm and AprioriTid algorithm. The algorithm first scans the database using the Apriori algorithm, then calculates the pruned database size, and then uses AprioriTid when its memory allows in order to find all frequent sets.

The current improvements of the algorithm are based on the following techniques:

(1) Hash (hash) table-based item set counting technique: the hash function is used to map each corresponding item set to a different column of the hash table, compare the column number with the supmin size, and eliminate the item set whose number of columns is lower than the supmin. Applying the hash technique can effectively reduce the number of candidate sets that need to be retrieved.

(2) Technique based on transaction compression: the transaction is also unlikely to contain any $k+1$-item sets, so such transactions can be directly eliminated or marked during statistics, thus reducing the number of transactions, that is, the technique based on transaction compression.

(3) Techniques based on dividing data: This technique requires only two scans of the database. The first scan is to divide the data into multiple parts and find the local frequent set of each part. Any frequent set in the dataset is a local frequent set and appears in at least one part of the divided data, as can be proved. The second scan is to evaluate each candidate set, solve for its support in the whole database, and find the global frequent set.

(4) Sampling-based technique: A nonempty subset of the given data is selected, with the subset size depending on the memory, and the frequent sets in the subset are mined using the Apriori algorithm. The supmin can be reduced moderately to avoid missing frequent sets.

(5) Dynamic item counting: This technique is proposed when the database is divided into chunks for mining. The main idea is to divide the database into data blocks, each of which can be mined, and then new candidate sets can be added to any of the data blocks. It is a dynamic technique since it needs to count the support of all item sets up to the current time.

2.4. FP-Growth Algorithm. The FP-growth algorithm [7, 8] is a classical algorithm proposed by Jiawei Han et al. This algorithm can complete the mining of association rules without generating candidate sets. The main idea is that the information in the transaction database is compressed into a frequent pattern tree FP-tree, and then the FP-tree is traversed, so that the required association rules can be obtained. The advantage of this algorithm is that the data in the transaction database correspond to the FP-tree one by one, and the operation of traversing the FP-tree replaces the operation of scanning the database, which reduces the I/O overhead and improves the efficiency of the algorithm.

2.4.1. FP-Tree-Related Definitions and Properties

Definition 1. FP-tree Frequent Pattern Tree [9]

An FP-tree is a tree structure that meets the following criteria: (1) There is one and only one root node with a null value, at least one item prefix subtree, and one frequent item header table. (2) Each node of the item prefix subtree must contain the three fields item name, count, and node pointer (node link). The significance of every field is item name is the identifier of the item that the node represents. Count represents the total number of transactions that reach the node's subpath. If the node does not exist, the value of node link is null; (3) The frequent item header table must contain the frequent item identifier and a header pointer to the first frequent item node in the tree with that identifier.

Property 1. For any frequent item, it is possible to obtain all frequent patterns containing that item by traversing the node chain of that item, beginning with the header pointer corresponding to that item in the frequent item header table.

If there is only one path in the frequent tree, then the set of all frequent items in the frequent tree can be obtained by enumerating the combination of nodes in the path, and its support is the minimum support of the combination of nodes in the path [10].

2.4.2. FP-Growth Algorithm. In order to reduce the number of input/output operations, the FP-growth algorithm introduces the compressed FP-tree data structure. The construction of the FP-tree is a crucial step in the FP-growth algorithm. This step involves scanning the transaction database and mapping each transaction to a path in the FP-tree until the final transaction has been mapped, at which point the FP-tree construction is complete. Different transactions may contain multiple identical items; consequently, their paths may partially overlap, and the greater the overlap, the greater the compression achieved with the FP-tree [11]. If the FP-tree is small enough to be stored in memory, the set of frequently occurring items can be extracted directly from memory [12], eliminating the need to repeatedly scan the database.

Construction of the FP-tree begins with the creation of a table of frequent item headers. Assuming a sup min of 20%, the first database scan obtains the number of occurrences of all 1-item sets, and the infrequent item sets in the 1-item set are removed based on the set sup min. Then, the frequent headers table is generated by sorting the remaining 1-frequent sets in descending frequency order. After generating the table of frequent headers, the database must be rescanned. For each individual record in the original data, for example, the first data record contains the items A, B, C, E, F, and O, where O is a rare occurrence. By removing O from this record, it becomes A, B, C, E, and F. As shown in Figure 1, this is sorted to yield the final result of A, C, E, B, and F. This is a concrete implementation [13, 14].

After getting the frequent item header table and the sorted data set, the FP-tree is built by reading the sorted dataset item by item, and then inserting the FP-tree in the

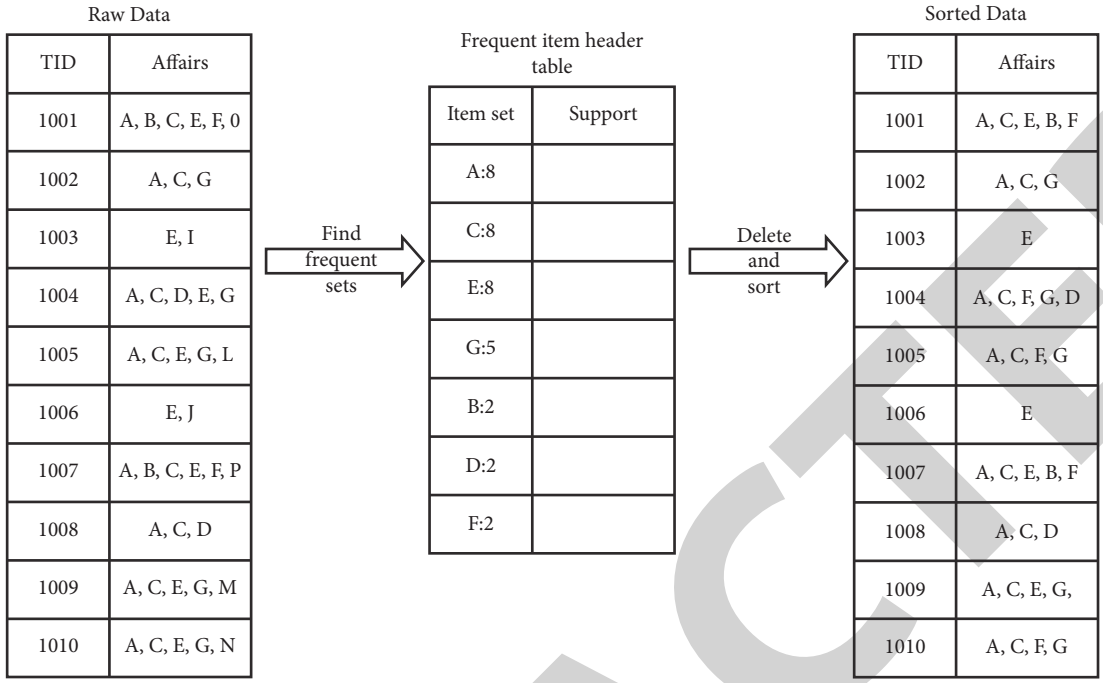

Figure 1: Preliminary data preparation diagram for FP-tree construction.

sorted order, where the ancestor node is the first one in the sorted order, and the descendant node is the last one in the sorted order. If a new node appears after insertion, the node corresponding to the item header table will be connected to the new node through the node chain until all data are inserted into the FP-tree, completing the establishment of the FP-tree [14].

First insert the first data set {*A*, *C*, *E*, *B*, *F*}, at this time the FP-tree tree has no nodes. Therefore, *A*, *C*, *E*, *B*, *F* is an independent path, and the count of each node is 1. The frequent item header table is linked to the corresponding new node through the node chain table, as shown in Figure 2. Then the second dataset {*A*, *C*, *G*} is inserted. Since the paths in *A*, *C*, *G*, and FP-tree have common ancestor nodes *A* and *C*, it is sufficient to add the new node *G*. Therefore, the total number of nodes in *G* is recorded as 1. The node count of *G* is therefore recorded as 1, while the node counts of *A* and *C* are changed to 2 by adding 1, and the node chain table is updated. As depicted in Figure 3, this is the outcome.

The completed FP-tree is obtained by inserting all the data according to the above steps, as shown in Figure 4 it is the result.

After the FP-tree is constructed, the next step is to mine the frequent set. According to the FP-tree, the frequent item header table and the node chain table, the conditional pattern base of each item in the frequent item header table and the FP-tree can be found by mining up from the bottom of the frequent item header table. The so-called conditional pattern base is the subtree of the target-mined node as a child node in the FP-tree. By eliminating the nodes in this subtree whose node counts are less than $\sup_{\min}$, the frequent item set can be recursively mined. Starting from the bottom node *F*, its corresponding conditional pattern base is {*A* : 8, *C* : 8, *E* : 6, *B* : 2, *F* : 2}. With this conditional pattern base, the L2s of *F* are obtained: {*A* : 2, *F*:2}, {*C* : 2, *F* : 2}, {*E* : 2, *F*:2}, {*B* :2, *F* : 2}. Combining these L2 sets yields L3: {*A* : 2, *C* : 2, *F* : 2}, {*A* : 2, *E* : 2, *F* : 2}, etc. Carrying on such recursive combinations, the maximum L5: {*A* : 2, *C* : 2, *E* : 2, *B* : 2, *F* : 2} is obtained. The subsequent frequent set mining of other nodes is similar to the above *F* node, and all frequent sets can be mined according to the above process.

The implementation of FP-growth algorithm is summarized as follows:

(1) Scan the data, delete the infrequent 1 item set, and generate the frequent item title table from the frequent item set, and sort the items in descending order.

(2) Scan the data again, remove the nonfrequent items from each transaction in the original data, and then sort the items in each transaction in descending order according to the support count of the item set.

(3) Read in the sorted data item by item and insert it into the FP-tree to complete the creation of the FP-tree.

(4) Find the conditional pattern base of each item from bottom to top according to the table of frequent project titles, and then find all frequent sets by recursively combining each conditional pattern base.

3. Method

3.1. Negative Association Rule Mining Process. The process of negative association rule mining consists primarily of the following steps: locating all frequent sets in the transactional dataset, for which different algorithms have distinct processing procedures, is the initial step. In this mining

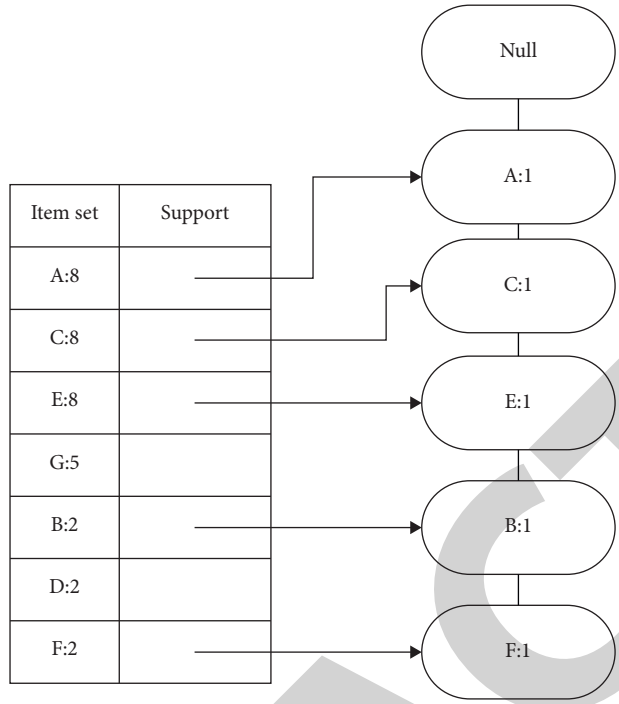

Figure 2: One-path FP-tree.

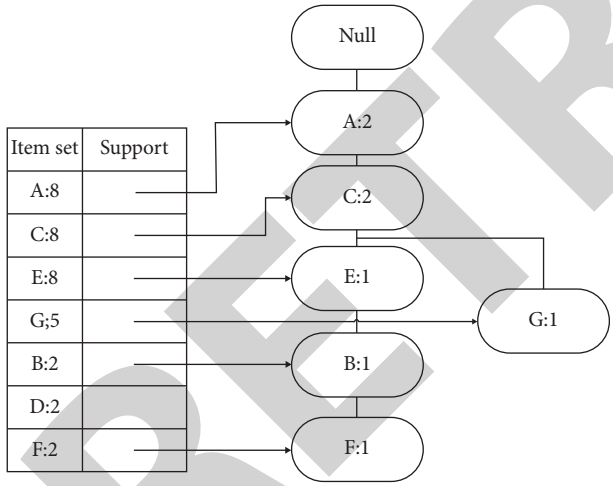

Figure 3: Two-path FP-tree.

procedure, support and trust are also indispensable factors. Due to the introduction of the negation property of the item set, however, more frequent sets are added, resulting in numerous redundant item sets, which complicates the mining algorithm and reduces mining efficiency [15]. Reducing the algorithm's complexity, enhancing the efficiency of mining, and discovering meaningful negative association rules.

3.2. Limitations of Association Rule Evaluation. At present, "support-confidence" as the evaluation criterion of association rules can effectively reduce the redundant item sets and meaningless association rules in the mining process. However, in the practical application of association rule mining, the redundancy level is still very high. Therefore, many invalid or useless association rules need to be further eliminated. Suppose a supermarket customer transaction data, as shown in Table 1.

Let M denote milk and N denote coffee. Assuming that $\sup_{\min}$ is 0.3 and conmin is 0.6, according to the data in Table 1, the support of the association rule "buy milk => buy coffee," that is, the support of the association rule "$M => N$," is $400/1000 = 0.4$ and its confidence $(M => N) = \text{support}\ (M => N)/(600/1000) = 0.67.400/1000 = 0.4$, and its confidence $(M => N) = \text{support}\ (M => N)/(600/1000) = 0.67$. The support degree of "$M => N$" is greater than 0.3 and the confidence degree is greater than 0.6, so it is a strong association rule, which means that customers are likely to buy coffee after buying milk. Therefore, it is a strong association rule, which means that customers are likely to buy coffee after buying milk. However, the support of the association rule "not buying milk => buying coffee" is $350/1000 = 0.35$, and its confidence level is $0.35/(400/1000) = 0.875$, which means that the probability that a customer will buy coffee after not buying milk is greater than the probability that a customer will buy coffee after buying milk. This situation of conflicting positive and negative association rules cannot be solved by relying only on the two threshold parameters of support and confidence for evaluation. We need to introduce the correlation degree into the original evaluation criteria of association rules and form the evaluation system of "support degree-confidence degree-correlation degree" [16].

Definition 2. Correlation between sets of items

Suppose there exist item sets I, A, B, and both A and B are contained in I, that is, $A \subseteq I$, $B \subseteq I$, D are the databases in which

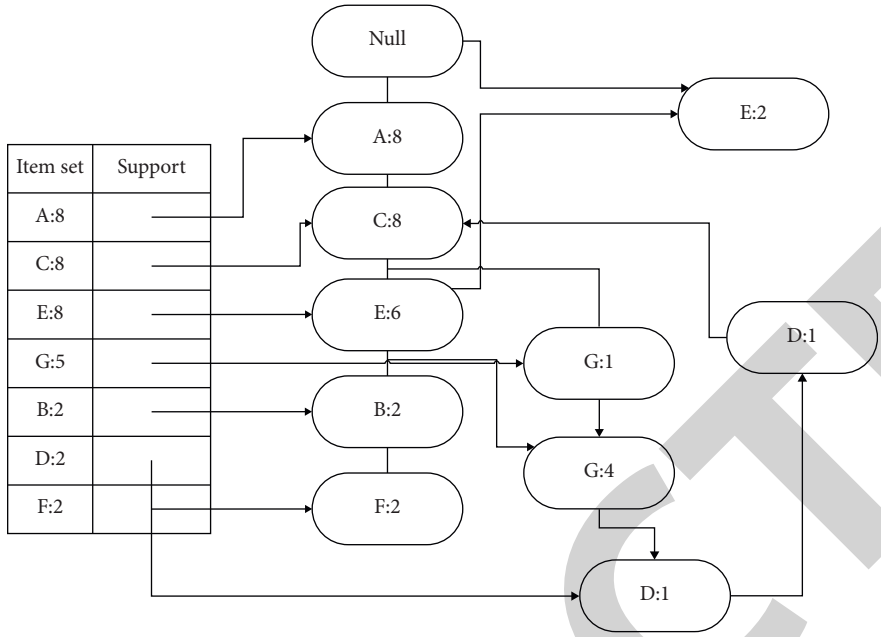

Figure 4: Complete FP-tree.

Table 1: Transaction data of a supermarket.

	Buying milk	Not buying milk	Total
Buy coffee	400	350	750
Do not buy coffee	200	50	250
Total	600	400	1000

item set A and item set B exist or occur, then the correlation between item set A and item set B is shown in (1) [17].

$$\text{relevance}(A, B) = \frac{P(A \cup B)}{P(A)P(B)}. \quad (1)$$

3.3. Update of Association Rules

3.3.1. Overview of Association Rule Updates. The majority of association rule mining research focuses on static databases or databases with unchanging data records. In practical applications, however, no database is static. As the number of database operations (additions, deletions, and modifications) increases, the database's data records also change; this is a dynamic rather than a static process, such as system logs that increase as user operations increase [18]. Determining the most recent association rules that reflect the current state of the database in order to provide a solid data foundation for user decisions necessitate an efficient association rule to update algorithm. The update algorithm is utilized to discover valid knowledge based on previously mined old data and unmined new data. Nonetheless, it is quite complex. Because invalid information in the old data may become valid information when combined with the new data, and valid information in the old data may become invalid information after a portion of the old data is removed, and mining association rules are a very complex process.

The preceding algorithms, including FUP, assume that data records change while sup min and con min remain unchanged. Nonetheless, when the data record remains unchanged and sup min or con min is modified, there are additional considerations to make when updating the association rules. There are two primary cases [19]: first, sup min remains unchanged while con min is modified, and second, sup min is modified while con min remains unchanged. The first case is straightforward to calculate because it has no impact on locating frequent items. In the second case, when sup min. becomes large, it is only necessary to eliminate the frequent items that do not satisfy the minimum support after the change, and the calculation is not complicated; when sup min. becomes small, there is a high probability that the previously nonfrequent items will become frequent items, causing the number of frequent items to increase, and the increase of frequent items will cause the number of frequent sets to increase. Regarding this, Yucai Feng et al. proposed an Apriori-based IUA algorithm. However, IUA also has issues with excessive I/O overhead and candidate sets. In addition, IUA suffers from the absence of frequent sets, which results in incomplete data mining of association rules.

To meet the need for association rule updates, it is possible to save in advance useful results computed from the original data. However, the Apriori algorithm has a high I/O overhead, inflated candidate sets, and low algorithm efficiency when searching for all item sets. Using the FP-growth algorithm for computation will produce a large number of FP-trees, which impact the algorithm's efficiency. Neither of these options is suitable. Liu Jingchun et al. derived the QAIS algorithm from this. The algorithm is based on the integration item set, and the frequent set is uniformly managed. When sup minor con min is changed, it is not necessary to repeatedly scan the database and generate candidate sets;

```
(1)  scan DB scan DB
(2)  for each transaction T ∈ DB {
(3)  find all subitem of T subItemSet
(4)    for i = 1; i < max_t; i++ {
(5)      if length of subitem = i {
(6)        if i_hashTable not contain the subitem {
(7)          subitem.count = 1
(8)          add subitem to i_hashTable
(9)        } else {
(10)         subitem.count + = 1
(11)         value = subitem.count
(12)         update value of key subitem in i_hashTable
(13)         }
(14)       }
(15)     }
(16)   }
(17)  If there is an update to the database data, cache the updated data at the same time when updating the database data
(18)  Cache the updated data to the dataset db_delete, db_add, db_update, db_noupdate, respectively
(19)  if db_update not null {
(20)    add db_update to db_add
(21)    add db_noupdate to db_delete
(22)  }
(23)  if db_delete not null {
(24)  for each transaction T ∈ db_delete {
(25)    find all subitem of T subItemSet
(26)      for each subitem ∈ subItemSet {
(27)        find the subitem key-value
(28)        subitem.count - = 1
(29)        update value of key subitem in i_hashTable
(30)        }
(31)      }
(32)  }
(33)  if db_add not null {
(34)  for each transaction T ∈ db_add {
(35)  Repeat operations (3), (4) and (5)
(36)  scan 1_hashTable to max_t_ hashTable, and find all frequent sets according to sup_min
(37)  Generate strong association rules based on the correlation of items in the frequent set and con_min, and eliminate the redundant
      association rules
```

Algorithm 1: Our algorithm.

rather, it is sufficient to filter the frequent set in the integration item set based on sup min, and then generate strong association rules based on the frequent set and con min. Thus, the QAIS algorithm is superior to Apriori and FP-growth for synchronizing association rules with data updates, as it avoids their flaws.

3.3.2. QAIS. The AIS algorithm [20] is a search-based fast association rule algorithm. The main idea of this algorithm is to scan the transaction database and find the set of nonempty subsets of items for each transaction, which is called SubItemSet. Then all the SubItemSet are integrated to get the integrated item set, which is called AggItemSet, and the AggItemSet is traversed to filter out the infrequent sets based on the minimum support to get all the frequent sets. Finally, a meaningful association rule is generated from the frequent set and the given con_{min}. The algorithm only needs to scan the database once to get an AggItemSet containing support counts, replacing the original database with AggItemSet, which reduces I/O overhead, does not need to generate candidate sets to get all frequent sets, and improves the efficiency of finding frequent sets. The following is an example of this algorithm.

Assuming that supmin is 0.4, the minimum support count is 4 for this original data. According to the QAIS algorithm, the original data is scanned, all nonempty subsets of each record are solved, and AggItemSet is obtained after integration, and then all frequent sets are solved according to supmin.

3.4. Our Algorithm. In addition to integrating frequent sets into the transaction database, our algorithm manages frequent sets uniformly. Similar to the QAIS algorithm, our algorithm discards the data structure of SubItemSet and modifies the data structure of AggItemSet by replacing the single-linked table with a hash table. A hash table is a data

structure whose elements are key-value pairs that are accessed based on the key code values. The linking order of the pointers in the chain table determines the logical order of the elements in a linked table [22]. Therefore, excluding the required data, additional space is required to store the pointer information, resulting in low-space utilization and storage density of the data structure. In addition, when querying data in the linked table structure, it is necessary to begin at the head of the linked table and query in reverse order, which is inefficient compared to the sequential table structure. Linked tables are superior to sequential tables when inserting and deleting data. The performance of insertion and deletion decreases with the number of table elements. A linked table, on the other hand, performs a comparison operation to determine where to insert data without moving the table's elements. The hash table, on the other hand, combines the benefits of both, enabling both fast querying and fast insertion and deletion without requiring the storage of additional overhead such as pointers.

Our algorithm caches updated data; that is, it caches the data as it is added, deleted, or modified. The cached data can then be processed in a specific amount of time based on the frequency of data updates and the volume of updated data, and new association rules can be mined. Consequently, the algorithm must consume additional space to store the updated data, and the amount of space required can be determined by analyzing the specific problem.

Suppose the original database is DB, db_delete is the deleted data in DB, db_add is the added data, db_update is the modified data in the original DB, and db_noupdate corresponds to db_update, which is part of the original data before being modified.

Our algorithm is described as follows:

Input: transactional database DB, minimum support $\sup_{\min}$, minimum confidence $\text{con}_{\min}$, maximum transaction length max_*t*.

Output: nonredundant association rules.

Step. Scan the transaction database DB, find all nonempty subsets for each transaction, and store each nonempty subset in different hash tables according to the item set length, such as 1-item set in one hash table and 2-item set in another hash table. The key-value pairs in the hash table store the item sets and their support counts separately. When the records in the database DB are changed, the corresponding records are cached at the same time. The added records are stored in the dataset db_add, the deleted records are stored in the dataset db_delete, the changed records are stored in the dataset db_update, and the original records corresponding to the changed records are stored in the dataset db_noupdate. To avoid wasting system resources by performing association rule mining for every updated record in the database, a specific time limit is set for processing the updated data. After determining the processing time limit, for the data sets that are not empty, all the nonempty subsets of each transaction in each data set are found and stored in the corresponding hash table. Finally, the valid strong association rule is found based on the correlation between $\sup_{\min}$ and $\text{con}_{\min}$ and the item set.

4. Intelligent System Design

4.1. Architecture. The system architecture consists of five core layers and three support systems. The five layers are user layer and operation layer and application layer and system layer, and foundation layer. The three major systems are infrastructure, integrated operation services, and system security. As show in Figure 5 it is the specific architecture.

4.2. Analysis of Data Processing

4.2.1. Analysis of Data. By analyzing the obtained service data, it is found that the popularity of community restaurants is usually related to the number of user evaluations, publicity and other factors, and the community restaurants are located in geographically dispersed areas, and the number of user evaluations of restaurants is distributed in a descending order with the overall popularity, as shown in Figure 6.

4.3. Recommended Results. To determine the efficacy of the recommendation system described in this paper, the intelligent recommendation with multiple minimum support weighted association rules, the user satisfaction, accuracy, and recall of the recommended services are evaluated. To compare different numbers of services, the system must establish distinct transaction databases for rule updates; rule updates execute the weighted association rule mining algorithm to replace previously generated rules based on all transactions in the transaction database, and rules are updated every 100 access records. At the start of the recommendation phase, no rules have been generated, so the first 100 transactions are recommended categorically. The rule-based smart recommendations cannot be utilized until the rules have been derived from the smart recommendations. This experiment is conducted by setting the number of services to 10, 20, and 50, and the results are displayed in Tables 2–5.

As can be seen, when the number of services is 10, there is no clear advantage of wisdom recommendation over traditional recommendation due to the small number of transactions in the 1st and 2nd tests; however, as the number of transactions increases in the later tests, the user satisfaction of wisdom recommendation increases, whereas traditional recommendation does not change significantly, indicating that the rules generated by the weighted association rule minimization algorithm are more effective. This indicates that the rules generated by the weighted association rule-mining algorithm play a role in the attraction recommendation, resulting in a more accurate recommendation. Such a trend is confirmed in tests involving 20 and 50 quantities, where the change is more pronounced and user satisfaction rises rapidly. To determine the measurement results, the precision and recall of the recommendations were computed. The average accuracy and recall are

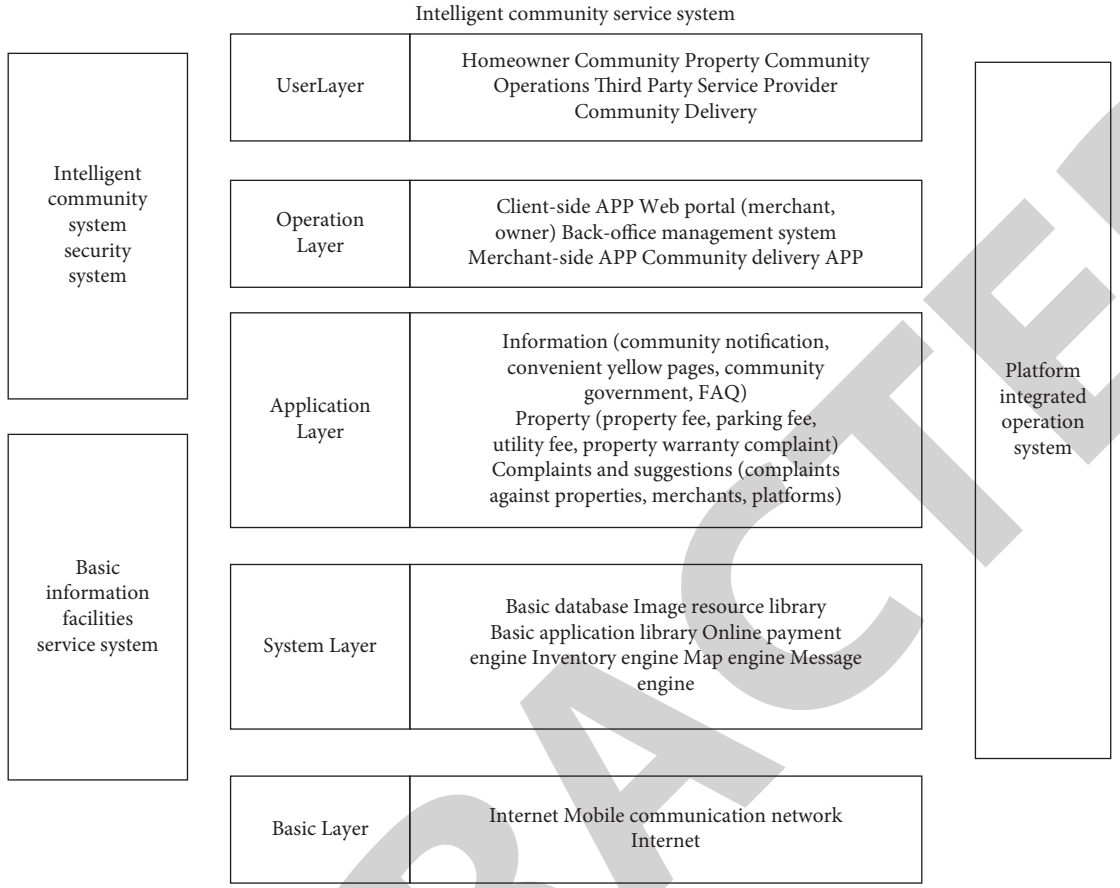

Figure 5: Overall system architecture diagram.

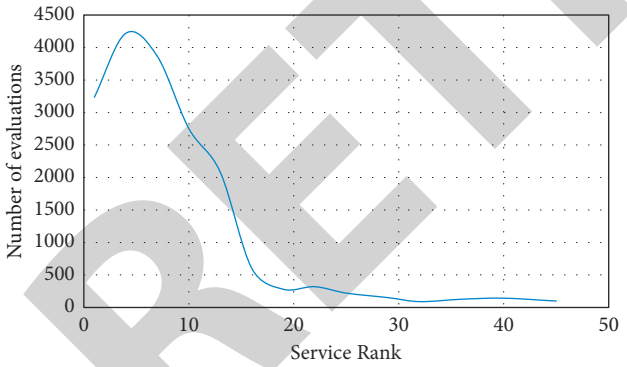

Figure 6: Number of service evaluations around the community.

Table 2: Test results for a number of sites of 10.

Test number	Number of transactions	Satisfaction recommended by this wisdom	Satisfaction of traditional association rule recommendation
1	001–200	4.2/138	4.5/152
2	201–400	4.1/161	4.2/135
3	401–600	4.7/155	4.3/103
4	601–800	4.6/192	4.4/170
5	801–1000	4.7/237	4.4/158

Table 3: Test results for a number of sites of 20.

Test number	Number of transactions	Satisfaction recommended by this wisdom	Satisfaction of traditional association rule recommendation
1	001–200	4.2/282	4.4/188
2	201–400	4.2/248	4.3/271
3	401–600	4.7/183	4.3/159
4	601–800	4.6/185	4.5/263
5	801–1000	4.7/252	4.3/179

Table 4: Test results for a number of sites of 50.

Test number	Number of transactions	Satisfaction recommended by this wisdom	Satisfaction of traditional association rule recommendation
1	001–200	4.2/127	4.4/142
2	201–400	4.1/235	4.2/231
3	401–600	4.6/297	4.3/229
4	601–800	4.6/195	4.2/129
5	801–1000	4.7/265	4.6/288

Table 5: Test results.

	Smart recommendation (%)	Recommendation by traditional association rules (%)
Accuracy	97.12	93.56
Recall	93.32	90.33

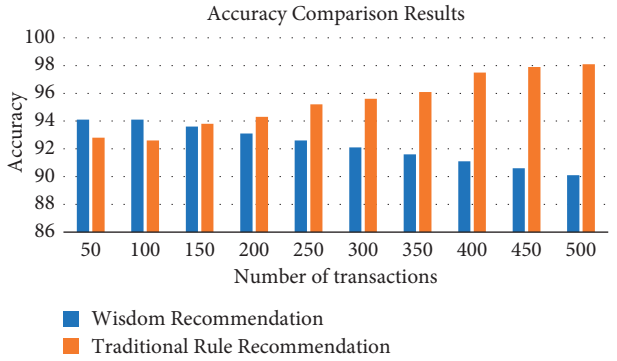

Figure 7: Accuracy comparison results.

displayed in Table 5, and Figure 7 compares the accuracy of the two algorithms.

5. Conclusion

In this paper, on the basis of the original smart city hardware devices, the data of take-out stores, gourmet restaurants, food markets, and supermarkets in smart communities are selected as the main research objects, and, based on association rules, data analysis is performed on the dietary information of community residents using the smart community platform in order to discover the purchasing habits and inherent similarities between customer groups. The system's recommendation accuracy has vastly improved, and it is a personalized recommendation method that can learn the special needs of small groups while simultaneously learning the basic needs of the general public, and these special needs are reflected in various aspects. The design of the weighting method becomes an integral component of the system's overall architecture. According to the characteristics of smart communities, we base the user access operation behavior data and introduce the user interest degree as a parameter to measure the operation behavior data, forming the basic data for data mining. Compared with the classification recommendation method, the user is able to locate the community information they are interested in very easily under the guidance of the service recommendation, and this personalized recommendations bring about a significant increase in user satisfaction [21, 22].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that he has no conflicts of interest.

References

[1] L. Zhang, K. Lai, and M. Liu, "The basic concept and theoretical system of intelligent tourism [J]," *Journal of Tourism*, vol. 27, no. 5, 2012.

[2] X. Tang, "Information technology to promote the transformation and upgrading of China's tourism industry[J]," *Business Times*, vol. 11, no. 25, 2010.

[3] H. Zhang, Z. Pan, and Y. Zhang, "Research and design of intelligent recommendation system for tourism services [J]," *Microcomputer Information*, vol. 22, no. 5-3, pp. 170-171, 2006.

[4] X. Wang, C. Mei, and X. Li, "Research and design of constraint-based travel recommendation system[J]," *Computer Technology and Development*, vol. 22, no. 2, pp. 141–145, 2012.

[5] J. Tao, "Design and implementation of a distributed intelligent recommendation system [J]," *Computer Simulation*, vol. 24, no. 7, pp. 296–300, 2007.

[6] J. Du and Y. Zhou, "Study on data-based tourism management decision support system," *Acta Automatica Sinica*, vol. 35, no. 6, pp. 834–840, 2009.

[7] J. Lin and J. P. Du, "Intelligent travel itinerary navigation system [J]," *Computer Applications*, vol. 23, no. 5, pp. 334–338, 2009.

[8] G. Büyüközkan and B. Ergün, "Intelligent system applications in electronic tourism," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6586–6598, 2011.

[9] H. H. Owaied, H. A. Farhan, N. Al-Hawamde, and N. Al-Okialy, "A model for intelligent tourism guide system," *Journal of Applied Sciences*, vol. 11, no. 2, pp. 342–347, 2011.

[10] Q. Liao, C. F. Hao, and C. H. Chen, *Data Mining and Mathematical Modeling [M]*, Vol. 22, National Defense Industry Press, Beijing, 2010.

[11] J. Bi and Z. Zhang, "A review of association rule mining algorithms [J]," *China Engineering Science*, vol. 7, no. 4, pp. 88–94, 2005.

[12] F. Zeng and X. Zhang, "App lication of cluster analysis to preventive maintenance scheme design of pavement[J]," *Ournal of H Arbin Institu Te of Technology*, vol. 16, no. 4, pp. 581–586, 2009.

[13] J. Sheng-Yi, L. Xia, and Z. Qi, *Data mining principles and practices [M]*, Vol. 51, Electronic Industry Press, Beijing, 2011.

[14] Y. C. Liil, T. P. Hong, and W. Y. Lin, "Mining association rules with multiple minimum supports using maximum constraints [J]," *International Journal of Approximate Reasoning*, vol. 40, no. 7, p. 44, 2005.

[15] M. C. TsengTseng and W. Y. Lin, "Efficient mining of generalized association rules with non-uniform minimum support," *Data & Knowledge Engineering*, vol. 62, no. 1, pp. 41–64, 2007.

[16] K. Wang, Y. He, and J. Han, "Pushing support constraints into association rules mining[J]," *Knowledge and Data Engineering*, vol. 15, no. 3, pp. 642–658, 2003.

[17] A. J. Lee, W. Lin, and C. S. Wang, "Mining association rules with multi-dimensional constraints," *Journal of Systems and Software*, vol. 79, no. 1, pp. 79–92, 2006.

[18] W. Ouyang, Z. Cheng, and Q. Cai, "The discovery of weighted association rules in databases [J]," *Journal of Software*, vol. 12, no. 4, pp. 612–619, 2001.

[19] C. Li and T. Yang, "An improved method for weighted association rule mining [J]," *Computer Engineering*, vol. 36, no. 7, pp. 55–57, 2010.

[20] B. Du, W. Li, and H. Shi, "A new fuzzy weighted association rule mining algorithm [J]," *Computer Engineering*, vol. 34, no. 20, pp. 218–220, 2009.

[21] C. Xie, M. Wen, L. Li, and Y. Li, "Theoretical study on transformation and upgrading of tourism industry[J]," *Journal of Liaoning Normal University (Natural Science Edition)*, vol. 33, no. 1, 2010.

[22] Z. Chen, H. Zhang, J. Ge, and H. Zheng, "Related literature recommendation based on weighted association rule mining [J]," *Intelligence Analysis and Research*, vol. 5, no. 10, pp. 57–61, 2007.