Blind identification of network protocols based on improved Apriori algorithm

Yingchun Chen*, Yuchen Dai*, Jingliang Xue and Fang Dong

Information Engineering University, Zhengzhou, Henan, 450002, China
*springer_2002@163.com, dyc_contact@163.com

Abstract. With the development and application of network technology and private protocols, more and more unknown protocols are found in communication networks. Analysis and identification of unknown protocols has become an important and difficult problem in obtaining valuable information from data. Based on the traditional Apriori algorithm, a reduced bit string algorithm and a weight compression algorithm are proposed to identify the unknown protocols according to the characteristics of network unknown protocol data in this paper. These proposed algorithms make use of location information, matrix compression, bit string and weight compression to gradually improve the identification accuracy and the processing efficiency of unknown protocols. Experiments on simulation data and actual data demonstrate the performance of the improved algorithms in terms of identification accuracy and time efficiency.

1. Introduction

With the rapid development of Internet, a large number of private protocols and more types of network protocols have emerged in order to meet various needs of data interaction[1]. The analysis of these unknown protocols has become the key to obtain valuable information. The data of one type of network protocols will appear specific characters in fixed fields, and different types of protocols are different. These provides a basis for using association rules to analyze network protocols[2]. The actual data will be noisy, distorted or missing. But it is still an effective method to identify unknown network protocols by finding the regularity between frequently occurring fields in data.

Apriori algorithm, which is a classical association rule algorithm, was proposed in 1994[3,4]. It is widely used[5,6]. But when the amount of data is large, the efficiency of the algorithm decreases[7]. The FP-Growth algorithm improves the efficiency by scanning the database once. But the algorithm is prone to fail to get the correct frequent itemsets when the length of the data exceeds a certain range[5]. The technologies of transaction data compression[7-8], data partitioning[9], dynamic itemset counting[10], Boolean matrix and distributed parallel computing[11] are also used to improve the performance of the Apriori algorithm. But these methods are generally used separately, without advantage combination. In references [12-13], the compression matrix method is proposed, but the method does not take into account the location information between frequent itemsets. For network protocol data, the same item in different position is likely to represent different types of protocol[14].

Aiming at the characteristics of network protocol data, this paper firstly adds location information to the basic Apriori algorithm to improve the identification accuracy. Then, by synthetically analyzing the relevant ideas in references [7], [8] and [11], a reduced bit string algorithm is proposed to improve the time efficiency of processing network data. Finally, according to the compression matrix method...
in references [12] and [15], the weight compression algorithm is proposed, which improves the processing efficiency and the identification accuracy of unknown network protocols.

2. Improved algorithm based on Apriori

2.1 Adding location information

In order to distinguish two items with the same value but in different position, the original data is processed in this paper. According to the characteristic that a protocol field is composed of integer bits or bytes, the location information is added to the data, as shown in Equation (1).

\[ D_L = D + C \times 2^\xi \]  

where \( D_L \) is the processed data. \( D \) is the original data, and \( C \) is the column where \( D \) is located. \( \xi \) is a constant and \( 2^\xi > \max(D) \). The inverse transformation of Equation (1) is shown in Equations (2) and (3).

\[ \hat{C} = \left\lfloor D_L / 2^\xi \right\rfloor \]  
\[ D = D_L - \hat{C} \times 2^\xi \]  

where \( \hat{C} = C \). By Equations (2) and (3), the original value and location of an item in the transaction matrix of Apriori algorithm can be obtained.

In processing the actual data, if the data is cut incorrectly, the method of adding location information may identify the same protocol into two types. In order to reduce such errors, this paper further uses the position interval between data fields to identify the relationship between frequent itemsets. Therefore, the location information added in this paper includes not only the location of data fields in transactions, but also the occurrence frequency of location intervals between frequent itemsets to increase the identification accuracy of unknown network protocol.

2.2 The reduced bit string algorithm

The time cost of the basic Apriori algorithm is mainly to traverse the transaction database many times, generate a large number of candidate itemsets and process them repeatedly. Besides adding location information to the basic Apriori algorithm, the proposed reduced bit string algorithm further compresses and reduces the related operations of each iteration process. It can quickly obtain candidate itemsets and their support degree through the logical operation of bit string.

2.2.1 Reduction idea

(1) Matrix multiplication

In various improved methods of the basic Apriori algorithm, it is always a key to get frequent 2-itemsets[10-11]. By combining two sets of frequent 1-itemsets, counting the support degree of candidate sets and pruning by comparing with threshold, the method of obtaining frequent 2-itemsets greatly limits the algorithm efficiency[9]. In this paper, the candidate sets of frequent 2-itemsets and their respective support degrees are obtained by matrix multiplication in one step.

(2) Matrix compression

To reduce the computational complexity and improve the time efficiency of the basic Apriori algorithm, the transaction matrix is compressed according to the generation process of frequent k-itemsets.

Measure 1 When \((k+1)\)-itemsets are obtained from frequent k-itemsets, delete rows whose sum value is not more than \(k\) in the transaction matrix.

If the number of items with the value of "1" in a row is less than \(k\), the number of "1" in the AND operation with other rows must be less than \(k\). The row can be deleted directly[10].

Measure 2 In frequent k-itemsets, if the occurrence frequency of an item is less than \(k+1\), delete the corresponding column from the transaction matrix.
In frequent k-itemsets, the frequent (k+1)-itemsets will be computed only when the number of items for each subset is not less than \(k + 1\). By deleting columns with the occurrence frequency of an item less than \(k + 1\) in the transaction matrix, it has no effect on the algorithm results[10-11].

**Measures 3** If the number of elements in frequent k-itemsets is less than \(k + 1\), the search of frequent itemsets will be terminated[11].

(3) **Connection step reduction**

The basic Apriori algorithm can not avoid the generation of repetitive and infrequent itemsets when combining two sets to get new itemsets. Therefore, it requires multiple operations to select the appropriate new itemsets[9,11]. In this paper, we first compare whether the first \(k - 1\) items of two itemsets are the same before they are combined to reduce the generation of redundant itemsets.

(4) **Pruning step reduction**

The occurrence frequency of candidate itemsets is first compared with the threshold to confirm the frequency itemsets. Only the frequent itemsets are stored and processed. It can avoid storing redundant intermediate variables and improve the time efficiency.

(5) **Introducing bit string**

Observing the data characteristics of Boolean matrix in the transaction database, the occurrence of each item in the matrix can be represented by a "bit string", which is called "transaction bit string" in this paper. By performing logical operations on each item's bit string, the support degree of candidate itemsets can be quickly obtained[10-11]. Correspondingly, the occurrence of an item in frequent k-itemsets can also be represented by a "bit string"[10], which is called "item bit string" in this paper. Logical OR operations are performed on item bit strings to generate candidate sets for the next new itemsets. Perform logical AND operations on transaction bit strings to count the occurrence frequency of itemsets in the transaction database[12].

### 2.2.2 Algorithm steps.

**Step 1** Add the location information to the original data in the database, and convert the data to Boolean matrix.

**Step 2** Scan the data in the database, and execute **Measure 1** and **Measure 2** to get frequent 1-itemset \(item_{new}\).

**Step 3** Multiply \(item_{new}\) by its transpose matrix \(item_{new}_t\) to get the result matrix. Compare the value of lower triangular matrix in the result matrix with the corresponding support degree threshold. If the value is larger than the support degree threshold, the location information is stored and the frequent 2-itemsets are generated.

In \(item_{new}\), if the data in the first column indicates the occurrence of item \(A\) in a transaction. Then in \(item_{new}_t\), the data in the second row indicates the occurrence of item \(B\) in the transaction. If the result value of the first column of \(item_{new}\) multiplying by the second row of \(item_{new}_t\) is 1, it means that \(A\) and \(B\) exist in the transaction at the same time. The element sum of the corresponding result vector is the occurrence frequency of \(\{A,B\}\) in the transaction. So, the result matrix obtained by \(item_{new}\) multiplying by \(item_{new}_t\) includes not only the information of the next new frequent itemsets, but also the occurrence frequency of the new itemsets. It completes two steps of the basic Apriori algorithm in one step.

**Step 4** Perform **Measure 1** and **Measure 2** on the frequent k-itemsets (\(k \geq 2\)) to get the reduced itemsets matrix.

**Step 5** If \(k \geq 3\), perform the connection step reduction to get the candidate itemsets of the next new itemsets. Otherwise, go to **Step 6**.

**Step 6** Calculate the support degree for each candidate itemset by introducing bit string. If the support degree is larger than the threshold, frequent (k+1)-itemsets are obtained.

**Step 7** Perform **Measure 3** on the frequent (k+1)-itemsets. Otherwise, repeat **Step 4** to **Step 7**.
2.3 The weight compression algorithm

In order to reduce the time complexity of the basic Apriori algorithm, matrix compression is used in references [12-15] to improve the algorithm performance. However, these algorithms do not consider the processing time of repeated transaction items in transaction matrix. In the reduced bit string algorithm, the transaction matrix is compressed. But, the compression only occurs when frequent itemsets are generated each time. It is not used at the beginning of scanning the database. At the same time, the matrix multiplication in the reduced bit string algorithm is only used to get the frequent 2-itemsets. Subsequent operations no longer use matrix multiplication because of too many items. Therefore, a weighted compression algorithm is further proposed in this paper. Based on the reduced bit string algorithm, the idea of weighting matrix and compression matrix is used to scan the original database once in the algorithm. The frequent itemsets are obtained by matrix multiplication to improve the time efficiency of the algorithm.

Using the characteristics of Boolean matrix, the weighted compression algorithm counts the same transactions in the transaction matrix to generate the corresponding weight matrix[10,12-13]. By updating and optimizing the weight matrix, the algorithm efficiency is further improved.

2.3.1 Improvement idea. (1) Construct a Boolean matrix to represent transactions. Only valid and non-repetitive transaction information is left in the Boolean matrix. The repetitive number of each transaction serves as the weight of the transaction. Thus, the weight matrix is constructed.

(2) Perform logical AND operation on the column vectors of the Boolean matrix. The step of generating candidate itemsets is skipped and the frequent itemsets are generated directly. The computing time of connection and pruning steps is reduced. At the same time, by matrix multiplication and matrix reduction, the computing time of obtaining high-order frequent itemsets is greatly reduced.

2.3.2 Algorithm steps. Step 1 Scan the transaction database to establish a Boolean matrix and a weight matrix. In the Boolean matrix, each row represents a transaction, and each column reflects the occurrence of items in the transaction.

Step 2 Scan the Boolean matrix and calculate the occurrence frequency for all items. If the frequency of an item is not greater than the threshold, the corresponding column vector is deleted.

Step 3 Perform logical AND operation on each column of the Boolean matrix to get the middle result. The middle result is multiplied by the corresponding weights of each row in the weight matrix to get the final result. The sum of the final result is the occurrence frequency of the candidate 2-itemsets. If the frequency satisfies the threshold requirement, the corresponding column vector is saved. Otherwise, it is deleted. The new Boolean matrix \( D_1 \) represents the frequent 2-itemsets.

Step 4 Sum each row in \( D_1 \). If the result value of the sum value of each row multiplying by the corresponding weight in the weight matrix is less than the minimum value of the support degree threshold, the row vector and the corresponding weight are deleted from the corresponding matrix. Generate a new Boolean matrix \( D_2 \) and a new weight matrix.

Step 5 Perform logical AND operations on each column in \( D_2 \). Repeat the column operation of Step 3 and the row operation of Step 4 to generate the frequent k-itemsets \( L_k \) \((k \geq 3)\). When the total number of transactions in the weight matrix meets the threshold requirement, the algorithm terminates.

3. Experimental results and analysis

3.1 Simulation experiment and result analysis

According to the characteristics of network protocols, 14 types of protocol data are constructed by MATLAB in the experiment. Each row of data has 20 columns, and each type of data has fixed fields in its specific location, as shown in Table 1. In Table 1, only the values in fixed fields are listed. The values in other fields are randomly generated and meet the transparent transmission requirement.
There are 5 categories of data in Table 1: A, B, C, D and E. Each category is divided into several subcategories. The data values in fixed fields of each category are the same, but their locations are different in different subcategories. A2 and A3 are the same type of protocol data, but data segments are cut in different location.

| Type | Fixed field | Number |
|------|-------------|--------|
| A1   | 18 6 9 500  |
| A2   | 18 6 9 500  |
| A3   | 18 6 9 500  |
| B1   | 18 10 7 500 |
| B2   | 18 10 7 500 |
| B3   | 18 10 7 500 |
| C1   | 18 5 8 500  |
| C2   | 18 5 8 500  |
| C3   | 18 5 8 500  |
| D1   | 19 10 7 500 |
| D2   | 19 10 7 500 |
| D3   | 19 10 7 500 |
| E1   | 20 5 8 500  |
| E2   | 20 5 8 500  |

3.1.1 Performance evaluation without location information. To test the algorithm identification performance, we compared the experimental results of three algorithms in the experiment: the basic Apriori algorithm (test algorithm 1), the reduced bit string algorithm (test algorithm 2) and the weight compression algorithm (test algorithm 3). When inputting the data of the original transaction matrix without location information, the identification results of the three algorithms are shown in Table 2. As Table 2 shows, the three algorithms can only recognize five categories of data. They can not effectively distinguish different types of protocols with the same frequent itemsets but different item locations. In terms of computational efficiency, test algorithm 3 performs better than the other two test algorithms. The computational efficiency of test algorithm 1 is the lowest.

| Algorithm | Type | Frequent item | Number | Time(s) |
|-----------|------|---------------|--------|---------|
| 1         | 20   | 5 8           | 1496   |         |
| 2         | 19   | 10 7          | 1498   |         |
| 3         | 18   | 5 8           | 1500   | 0.0123  |
| 4         | 18   | 10 7          | 1499   |         |
| 5         | 18   | 6 9           | 1500   |         |
| 1         | 20   | 5 8           | 1598   |         |
| 2         | 19   | 10 7          | 1500   |         |
| 3         | 18   | 5 8           | 1499   | 0.0038  |
| 4         | 18   | 10 7          | 1500   |         |
| 5         | 18   | 6 9           | 1500   |         |
| 1         | 20   | 5 8           | 1500   |         |
| 2         | 19   | 10 7          | 1499   |         |
| 3         | 18   | 5 8           | 1500   | 0.0025  |
| 4         | 18   | 10 7          | 1499   |         |
| 5         | 18   | 6 9           | 1500   |         |

3.1.2 Performance evaluation without frequent item location interval. Through Equation (1), the location information is added to the three test algorithms. But, the location interval information between frequent items is not added. The experimental results of the three test algorithms are shown in...
Table 3. The identified protocol types and the number of corresponding protocol data are shown in Table 3.

From Table 3, it can be seen that the different protocol subcategories can be identified by adding the relevant location information into the algorithm. But, due to the absence of the location interval information between frequent items, the identified protocol types are 15. These algorithms cannot deal with the identification error caused by inaccurate data segment cutting. Excluding the identification error of A2 and A3, the protocol identification accuracy of the three test algorithms are shown in Figure 1. As Figure 1 shows, test algorithm 3 has a better performance than the other two test algorithms in protocol identification.

Table 3. Experimental results of three algorithms without frequent item location interval

| Algorithm | A1 | A2 | A3 | B1 | B2 | B3 | C1 | C2 | C3 | D1 | D2 | D3 | E1 | E2 | E3 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1         | 488| 491| 493| 486| 488| 482| 494| 496| 486| 489| 498| 483| 480| 491|
| 2         | 497| 497| 494| 490| 497| 497| 500| 499| 500| 492| 498| 494| 491| 500|
| 3         | 500| 495| 497| 500| 495| 495| 500| 498| 500| 497| 500| 493| 498| 497| 500|

Figure 1. Protocol identification accuracy of three algorithms without frequent item location interval

3.1.3 Performance evaluation with frequent item location interval. By adding the location interval between frequent items to the three test algorithms, the experimental results of the three algorithms are shown in Table 4.

Table 4. Experimental results of three algorithms with frequent item location interval

| Algorithm | A1 | A2/A3 | B1 | B2 | B3 | C1 | C2 | C3 | D1 | D2 | D3 | E1 | E2 | E3 |
|-----------|----|-------|----|----|----|----|----|----|----|----|----|----|----|----|
| 1         | 490| 982   | 488| 490| 484| 496| 498| 498| 490| 491| 500| 485| 481| 493|
| 2         | 500| 995   | 493| 500| 497| 498| 499| 500| 495| 496| 500| 496| 494| 500|
| 3         | 500| 992   | 500| 500| 495| 500| 500| 500| 493| 498| 500| 498| 500| 500|

From Table 4, it can be seen that these algorithms can accurately identify 14 protocol types. The identification error caused by inaccurate cutting of data segments is solved by adding the location interval information between frequent items. The identification accuracy of the three algorithms is shown in Figure 2. As Figure 2 shows, test algorithm 3 still has the best protocol identification performance. The protocol identification accuracy is higher than that without considering the location interval between frequent items. The experimental results show the effectiveness of the improved algorithm in this paper.
With different minimum support degree thresholds, the running time of the three test algorithms is shown in Figure 3.

As Figure 3 shows, the change trend of the running time of test algorithm 1 and test algorithm 3 is from obvious to gentle with the increase of the minimum support degree threshold. Test algorithm 2 is not sensitive to the minimum support degree threshold. When the minimum support degree threshold is more than 200, the computational efficiency of test algorithm 3 and test algorithm 2 is significantly higher than that of test algorithm 1. In terms of running time and efficiency, test algorithm 3 has the best performance among the three test algorithms.

3.2 Actual data and result analysis
In this paper, we use Wireshark to capture actual network protocol data. 6 types of protocol data are analyzed in the experiment, as shown in Table 5. The data field descriptions of the six protocol types are shown in Figure 4. For the SSDP protocol, some specific marker data fields can be replaced by specific numbers because the characters are long and of varying lengths in the data fields.
Table 5. Actual network protocol data

| Type       | Number |
|------------|--------|
| TCP        | 100    |
| DHCPv6     | 100    |
| SSDP       |        |
| discover   | 100    |
| alive      | 100    |
| ICMPv6     |        |
| in IPv6    | 100    |
| in IPv4    | 100    |

TCP

| 12 bytes | 8   | 0   | 45  | 8 bytes | 6 |

DHCPv6

| 12 bytes | DD  | 60  | 8 bytes | 17 | ----- | 546 | 547 |

SSDP: discover

| 12 bytes | 8   | 0   | 45  | 8 bytes | 17 | 12 bytes | 1900 | ----- | ST | MAN | MX |

SSDP: alive

| 12 bytes | 8   | 0   | 45  | 8 bytes | 17 | ----- | 60   | 58   |

ICMPv6 in IPv4

| 12 bytes | DD  | 60  | 8 bytes | 58 |

ICMPv6 in IPv6

Figure 4. Six network protocol data formats

Perform the three test algorithms on the actual data. The experimental results of the three algorithms are shown in Table 6.

Table 6. Experimental results of three algorithms for actual data

| Algorithm | TCP | DHCPv6 | SSDP: discover | SSDP: alive | ICMPv6 in IPv4 | ICMPv6 in IPv6 | average identification accuracy % |
|-----------|-----|--------|----------------|-------------|----------------|----------------|----------------------------------|
| 1         | 97  | 99     | 97             | 95          | 98             | 98             | 97.33                            |
| 2         | 98  | 100    | 98             | 98          | 98             | 99             | 98.50                            |
| 3         | 99  | 100    | 98             | 98          | 99             | 100            | 99.00                            |

As Table 6 shows, for the actual network protocol data, test algorithm 3 has the best protocol identification performance among the three test algorithms. In terms of algorithm efficiency, compared with test algorithm 1 and test algorithm 2, test algorithm 3 is one order of magnitude higher. The running speed of test algorithm 2 is only about twice as fast as that of test algorithm 1.

4. Conclusion

In this paper, based on the frequent itemsets, the location information is added to the basic Apriori algorithm to improve the algorithm performance. By matrix compression and weight compression, a reduced bit string algorithm and a weight compression algorithm are proposed respectively for blind identification of unknown network protocols. The experimental results on simulation and actual data show that the reduced bit string algorithm and the weight compression algorithm are superior to the basic Apriori algorithm in terms of the identification accuracy and the time efficiency. For identifying the network protocols without any prior knowledge, the weight compression algorithm has the best performance among the three algorithms, which shows the rationality and effectiveness of the improved algorithm.
References

[1] Zhu, Y.Y., Wu, J.F., Ming, Z. (2011) Research on intrusion detection based on network events and deep protocol analysis. Journal on Communications, 32: 171-178.

[2] Liang, R. Research and Implementation of Network Traffic Identification Model. (2010) University of Electronic Science and Technology of China.

[3] Mao, G.J., Duan, L.J. (2016) Principles and Algorithms of Data Mining. Tsinghua University Press, Beijing.

[4] Zhao, X.J., Sun, Z.X., Yuan, Y. (2016) An efficient association rule mining algorithm based on prejudging and screening. Journal of Electronics & Information Technology, 38: 1654-1659.

[5] Li, J.Z., Ye, X.J. (2007) State-of-the-art in Association Rules Mining Algorithms. In: National Academic Conference on Computer Technology and Application. Ningbo. pp. 204-208.

[6] Wang, X., Liu, Y.Q., Qi, F. (2017) Control method of Apriori based exploratory warfare simulation experiment. Systems Engineering and Electronics, 39: 917-923.

[7] Chen, A., Chen, N., Zhou, L.X., et al. (2006) Mining Technology and Application. Science Press, Beijing.

[8] Chen, M.S., Han, J., Yu, P.S. (2002) Data mining: an overview from a database perspective. IEEE Transactions on Knowledge & Data Engineering, 8: 866-883.

[9] Wang, W. Research and Improvement for Apriori Algorithm of Association Rule. (2012) Ocean University of China.

[10] Du, X.P., Ma, X.L., Tang, S.W., et al. (2002) A fast algorithm for mining of association rules. Computer Engineering and Applications, 38: 1-4.

[11] He, Y.S. Research and Application on the Technologies in Mining Association Rules. (2010) Nanjing University of Aeronautics and Astronautics.

[12] Miao, M.M., Wang, Y.Y. (2013) Research on improvement of Apriori algorithm based on matrix compression. Computer Engineering and Applications, 49: 159-162.

[13] Fu, S., Liao, M.H., Song, D. (2012) The improved Apriori algorithm based on compression matrix approach. Microelectronics & Computer, 29: 28-32.

[14] Wang, S.H. Analysis of Feature for Protocol in the Form of Stream Data. (2016) University of Electronic Science and Technology of China.

[15] Li, Z.L. Research and Application of Apriori Algorithm Based on Cluster and Compression Matrix. (2010) Soochow University.