Journal of Universal Computer Science, vol. 22, no. 6 (2016), 822-835
submitted: 30/11/15, accepted: 28/6/16, appeared: 1/6/16 © J.UCS

A Quick Method for Querying Top-*k* Rules from Class Association Rule Set

Loan T.T. Nguyen
(Faculty of Information Technology, Nguyen Tat Thanh University
Ho Chi Minh City, Vietnam
nttloan@ntt.edu.vn; nthithuyloan@gmail.com)

Ngoc-Thanh Nguyen*
(Division of Knowledge and System Engineering for ICT and Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
Department of Information Systems, Faculty of Computer Science and Management, Wroclaw University of Technology, Wroclaw, Poland
nguyenngocthanh@tdt.edu.vn; Ngoc-Thanh.Nguyen@pwr.edu.pl)

Bogdan Trawiński
(Department of Information Systems, Faculty of Computer Science and Management
Wroclaw University of Technology, Wroclaw, Poland
Bogdan.Trawinski@pwr.wroc.pl)

Abstract: Finding class association rules (CARs) is one of the most important research topics in data mining and knowledge discovery, with numerous applications in many fields. However, existing techniques usually generate an extremely large number of results, which makes analysis difficult. In many applications, experts are interested in only the most relevant results. Therefore, we propose a method for querying top-*k* CARs based on their supports. From the set of mined CARs that satisfy the minimum support and the minimum confidence thresholds, we use a QuickSort-based method to query top-*k* rules. The whole rule set is partitioned into two groups. If the number of rules in the first group is *k*, then the first group is the set of result rules. If the number of rules in the first group is greater than *k*, the second group is partitioned to find the remaining top-*k* rules. Experimental results show that the proposed method is more efficient than existing techniques in terms of mining time.

Keywords: Data mining, class association rules, top-*k* class association rules.
Categories: I.2, M.1

1 Introduction

In 2012, Fournier-Viger et al. proposed the TopKRules algorithm for mining top-*k* association rules [Fournier-Viger, 12a]. They used two thresholds: minimum confidence (*minConf*) and *k*. Rules whose confidences do not satisfy *minConf* are removed. The authors also used some techniques to prune the search space to reduce runtime. However, this algorithm cannot mine enough *k* association rules when *minConf* is large. Fournier-Viger and Tseng proposed an algorithm for mining top-*k*

* Corresponding author

non-redundant association rules [Fournier-Viger, 12b]. Like TopKRules, the proposed algorithm cannot mine enough top-*k* non-redundant association rules when *minConf* is large. Some non-redundant rules are ignored by the pruning scheme. A method for finding filtered-top-*k* association rules has been proposed [Geoffrey, 11].

Deng and Fang proposed the NTK algorithm for mining top-rank-*k* frequent itemsets [Deng, 07]. NTK uses a divide-and-conquer scheme and an early pruning technique. Quyen et al. proposed an improved algorithm, named *i*NTK [Le, 15]. *i*NTK uses the subsume concept to quickly determine itemsets with the same rank.

The above algorithms focus on mining top-rank-*k* frequent itemsets or top-*k* (non-redundant) association rules. They cannot be used for mining top-*k* class association rules (CARs). Methods for mining top-rank-*k* frequent itemsets cannot be used for mining association rules and TopKRules cannot be used for mining top-*k* CARs because the right-hand side of an association rule is any frequent itemset whereas the right-hand side of a CAR only contains class labels. Therefore, an effective solution for mining top-*k* CARs is necessary.

A naïve approach should sort all found rules and select the top-*k* rules with the highest supports. However, this is inefficient, especially when the number of rules is very large. Recently, an InsertionSort-based algorithm for finding top-*k* CARs has been proposed [Nguyen, 16]. It maintains a list of sorted *k* rules and continuously replaces rules at the end of the list with rules with higher supports from the remaining rule set. This algorithm is more efficient than that based on sorting when *k* is small. When *k* is large, the insertion-based method is not as efficient as the naïve technique because it must sort many rules in the first step.

In this paper, we propose an algorithm for mining top-*k* CARs from the mined rule set. Our algorithm uses a QuickSort-based method for effectively querying top-*k* CARs. The whole rule set is first partitioned into two groups. The first group contains rules whose supports are greater than or equal to that of a chosen rule *x* and the second group contains the rest of the rule set. If the number of rules (*kl*) in the first group is smaller than *k*, all of the rules in this group belong to the result set. Therefore, we only need to find $k - kl$ rules from the second group, which is done in the same way as processing the whole rule set. Otherwise, no rules in the second group belong to the result set, so we only need to find the results from the first group.

The rest of this paper is organized as follows. Section 2 presents works related to mining CARs and top-*k* frequent itemsets/association rules. The main contributions are described in Section 3. Section 4 presents the experimental results. Section 5 gives the conclusions and suggestions for future work.

2 Related Works

2.1 Mining Class Association Rules

There are many methods for rules-based classification. Breiman et al. [Breiman, 84] proposed a binary tree for mining rules. The CART (classification and regression tree) algorithm was also proposed. CART chooses the attribute to split data using Gini index measure. ID3 [Quinlan, 86] and C4.5 [Quinlan, 92][Do, 2015], two decision-tree-based approaches, have been proposed. ID3 and C4.5 use the information gain and the ratio gain to choose the attribute, respectively. ILA and ILA-2 [Tolun, 98a]

[Tolun, 98b], rules-based methods for prediction, have been proposed. Unlike CART, ID3, and C4.5, ILA and ILA-2 do not build a tree; instead, they find rules using maximum combination. In 2011, Parker analyzed seven ways to determine the best classifier in the set of classifiers [Parker, 11]. This method can be applied in any classifier.

Associative classification, which integrates association rule mining and classification [Liu, 98][Thabtah, 07a], is an efficient classification approach. A particular subset of association rules whose right-hand side is restricted to the class attribute is mined. This subset of rules is denoted as CARs.

The first method for mining CARs was proposed in 1998 [Liu, 98]. The CBA algorithm was developed in this work. CBA is based on the Apriori method for mining CARs and uses a heuristic method to build a classifier. In 2001, a frequent pattern (FP)-tree-based method for classification based on multiple association rules was proposed [Li, 01]. The authors modified the FP-tree for storing single items with their class information. CARs are then mined from the FP-tree and stored in a class rule (CR)-tree. To build a classifier, a database coverage threshold is used to select the rules. Classification based on predictive association rules [Yin, 03]. Thabtah et al. used multi-class, multi-label association classification to mine and predict the class of new records [Thabtah, 04][Thabtah, 05]. Thabtah and Cowling proposed a greedy method to build a classifier to predict the class of new records using multiple rules [Thabtah, 07b]. CAR mining based on the equivalence class rule (ECR)-tree was proposed by Vo and Le [Vo, 08]. The proposed algorithm (ECR-CARM) first scans the dataset to build the first level of the ECR-tree. It then expands the ECR-tree to build child nodes using the parent nodes. CAR-Miner and CAR-Miner-Diff, two improved versions of ECR-CARM, have been developed [Nguyen, 13][Nguyen, 15a]. Chen et al. proposed the principal association mining (PAM) method to improve the accuracy and size of the classifier [Chen, 14a]. Some efficient methods have been proposed to improve accuracy, such as methods that use CBA to handle class imbalance [Chen, 12] and uncertain datasets [HooshSadat, 12], methods that use interestingness measures [Lan, 06][Shaharanee, 11][Nguyen, 15b][Vo, 11], a method that uses rule prioritization [Chen, 14b], and a method that uses closed sets [Liu, 09]. However, none of these techniques is designed for finding top-k CARs.

2.2 Mining Top-rank-*k* Frequent Itemsets

Deng et al. proposed the NTK algorithm for mining top-rank-*k* frequent itemsets [Deng, 14]. NTK represents patterns with the Node-list data structure. It uses *t*-patterns to form (*t*+1)-patterns. By using Node-list, the algorithm does not need to rescan the dataset when computing the support of (*t*+1)-patterns. The main ideas of NTK are as follows:

1) NTK traverses the pre-order post-order code (PPC)-tree and generates a Node-list of 1-patterns. It then finds 1-patterns that belong to top-rank-*k* and inserts them into the top-rank-*k* table. This table contains frequent 1-patterns and their supports. All patterns with the same support are stored in the same entry. Therefore, the number of entries in this table is smaller than *k*.

2) 1-patterns in the result are used to generate candidate 2-patterns. NTK inserts candidate 2-patterns into the top-rank-*k* table if their support is not smaller than the smallest support of patterns in this table.

Step 2 is repeated using t-patterns in the top-rank-k table to create candidate (t+1)-patterns until no candidates can be generated.

[Le, 15] developed an improved algorithm, called iNTK, based on NTK. iNTK uses t-patterns to create candidate (t+1)-patterns. By using an N-list, it does not need to rescan the dataset to compute the support of candidate (t+1)-patterns. The algorithm uses the subsume concept to reduce the number of generated candidates compared to those for NTK, reducing the time required to generate candidates.

2.3 Mining Top-*k* Association Rules

Fournier-Viger et al. proposed the TopKRules algorithm for mining top-k association rules from datasets [Fournier-Viger, 12a]. This algorithm uses the *minConf* value during the mining process of top-k rules. The *minSup* value depends on the lowest support of itemsets. The TopKRules algorithm is based on the principle of extending rules and uses some methods for early eliminating rules that do not belong to top-k rules. Fournier-Viger and Tseng extended TopKRules for mining top-k non-redundant rules [Fournier-Viger, 12b] and top-k sequential rules [Fournier-Viger, 11].

3 Method for Mining Top-*k* Class Association Rules

3.1 Basic Concepts

Let D be the set of training data with n attributes A_1, A_2, …, A_n and $|D|$ objects (cases). Let $C = \{c_1, c_2, \ldots, c_k\}$ be a list of class labels. A specific value of an attribute A_i and class C are denoted by the lower-case letters a and c, respectively [Nguyen, 15a].

Definition 1: An itemset is a set of some pairs of attributes and a specific value, denoted $\{(A_{i1}, a_{i1}), (A_{i2}, a_{i2}), \ldots, (A_{im}, a_{im})\}$.

Definition 2: A CAR r is of the form $\{(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})\} \rightarrow c$, where $\{(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})\}$ is an itemset, and $c \in C$ is a class label.

Definition 3: The actual occurrence $ActOcc(r)$ of a rule r in D is the number of rows of D that match r's condition.

Definition 4: The support of a rule r, denoted $Sup(r)$, is the number of rows that match r's condition and belong to r's class.

Definition 5: The confidence of a rule r, denoted by $Conf(r)$, is defined as:

$$Conf(r) = \frac{Supp(r)}{ActOccr(r)}$$

For example, consider rule $r = \{<(A, a1)> \rightarrow y\}$ for the dataset in Table 1. We have:

$$ActOccr(r) = 3$$
$$Sup(r) = 2$$
$$Conf(r) = \frac{Supp(r)}{ActOccr(r)} = \frac{2}{3}.$$

Definition 6: Given a set of CARs R and a rule $r \in R$, the rank of r in R is defined as follows: $Rank(r) = |\{r_i \in R \mid Sup(r) > Sup(r_i)\}| + 1$.

Definition 7: (Top-*k* rules according to support): Given a set of CARs *R* and a threshold *k*, mining the top-*k* CARs is equivalent to finding the *k* best rules in *R* based on their supports, i.e.:

$$\text{Top-}k(R) = \{r \in R \mid Rank(r) \leq k\}$$

Based on the above two definitions, the problem of mining top-*k* CARs is simply to filter out *k* rules whose supports are highest.

OID	A	B	C	class
1	a1	b1	c1	y
2	a1	b2	c2	n
3	a2	b2	c1	n
4	a3	b3	c1	y
5	a3	b1	c2	n
6	a3	b3	c1	y
7	a1	b3	c2	y
8	a2	b2	c2	n

Table 1: Example of training dataset

Below is a QuickSort-based algorithm for mining top-*k* CARs from the mined rule set. The algorithm employs the idea of the QuickSort algorithm for partitioning the rule set into two groups. The first group contains CARs whose supports are greater than or equal to *x* (*x* is the support of a chosen rule). The second group contains CARs whose supports are smaller than *x*. Assuming that *kl* is the number of rules in the first group and *kr* is the number of rules in the second group, the following proposition is used to speed up the runtime.

Proposition 1: Assuming that the number of rules in the rule set is greater than or equal to *k*. We have:

1. if $kl > k$ then all rules in the second group cannot belong to top-*k* CARs.
2. if $kl < k$ then all rules in the first group belong to top-*k* CARs.
3. if $kl = k$ then the first group is the top-*k* CARs.

Proof:

1. $kl > k$ means that the number of rules in the first group is greater than *k*. All rules in the second group have supports smaller than *x* while all rules in the first group have supports greater than or equal to *x*, and thus none of the rules in the second group belong to the top-*k* CARs.
2. Because all supports of rules in the first group are greater than or equal to *x* and *x* is greater than all supports of the rules in the second group, the minimum support of the first group is always greater than the maximum support of the second group. Therefore, *kl* rules in the first group always belong to the top-*k* CARs. We need to find $k - kl$ CARs from the second group.

3. According to the above two proofs, all rules in the first group have supports greater than those of rules in the second group. The number of rules in the first group is k. Therefore, the first group is the result.

Based on Proposition 1, we propose an algorithm for mining top-k CARs, shown in Figure 1.

3.2 Algorithm

Algorithm 1.
Input: A set of CARs (R) and threshold k.
Output: Top-k CARs in R

```
QuickSort-Top-k(R, left, right, k)
Begin
1.  if (right = k)
2.        return;
3.  i = left;
4.  j = right;
5.  x = Sup(R(i+j)/2);
6.  while (i <= j)
    Begin
7.    while (Sup(Ri) > x) i = i + 1;
8.    while (Sup(Rj) < x) j = j - 1;
9.    if(i <= j)
    Begin
10.       Swap (Ri, Rj);
11.       i = i + 1;
12.       j = j - 1;
    End
End
13. if (i > k) QuickSort-Top-k(R, left, i-1, k);
14. else QuickSort-Top-k(R, i, right, k);
End
//Main function
Begin
15. QuickSort-Top-k(R, 1, |R|, k);
16. RS = ∅;
17. For (i = 1; i <= k && i <= |R|; i++)
18.     RSi = Ri;
19. Return RS;
End
```

Figure 1: Proposed QuickSort-based algorithm

Figure 1 describes the proposed algorithm for finding top-k CARs based on the partitioning idea of QuickSort. Line 1 compares *right* with k; if *right* = k, there is nothing to be done because all rules from R_1 to R_{right} belong to the top-k CARs. Lines 3 and 4 set i = *left* and j = *right*. x is the support of the middle rule (line 5). If the support of R_i is greater than x, then we increase i; if the support of R_j is lower than x, then we decrease j (lines 7-8). Line 9 compares the values of i and j; if i is lower than

j, then the two rules at positions *i* and *j* are swapped. After swapping, the value of *i* is increased and the value of *j* is decreased (lines 11-12). We repeat the above steps until *i* greater than *j*. If *j* is greater than *k*, then Quick-Sort-Top-*k*(*R*, *left*, *j*, *k*) is called recursively. Otherwise, Quick-Sort-Top-*k*(*R*, *i*, *right*, *k*) (lines 13-14) is called.

QuickSort-Top-*k*(*R*, 1, |*R*|, *k*) sorts the rule set based on QuickSort and gets enough *k* rules. On line 16, the algorithm assigns null for *RS*. Lines 17 and 18 are used to copy the result to *RS*. When the algorithm ends, it returns the set of found rules *RS*.

3.3 Example

Using CAR-Miner for the dataset in Table 1 with *minSup* = 20% and *minConf* = 80%, the MECR-tree is obtained as shown in Figure 2.

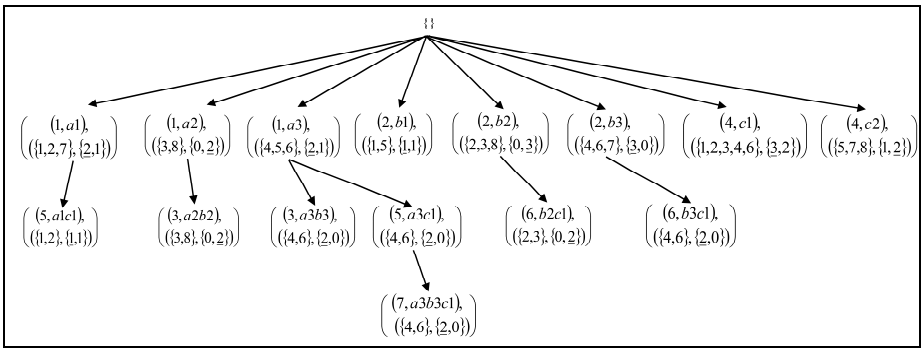

Figure 2: MECR-tree for dataset in Table 1

From Figure 2, we have 9 rules whose confidences satisfy *minConf*. They are shown in Table 2.

No.	R_i	Sup	Conf
R_1	If *A* = *a2*, then class = *n*	2/8	1
R_2	If *A* = *a2* and *B* = *b2*, then class = *n*	2/8	1
R_3	If *A* =*a3* and *B* = *b3*, then class = *y*	2/8	1
R_4	If *A* = *a3* and *B* = *b3* and *C* = *c1*, then class = *y*	2/8	1
R_5	If *A* = *a3* and *C* = *c1*, then class = *y*	2/8	1
R_6	If *B* = *b2*, then class = *n*	3/8	1
R_7	If *B* = *b2* and *C* = *c1*, then class = *n*	2/8	1
R_8	If *B* = *3*, then class = *y*	3/8	1
R_9	If *B* = *b3* and *C* = *c1*, then class = *y*	2/8	1

Table 2: Set of CARs from dataset in Table 1

From Table 2, we have *R*, which includes CARs with their supports, as follows:

R	**R_1**	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**	**R_7**	**R_8**	**R_9**
Sup	2/8	2/8	2/8	2/8	2/8	3/8	2/8	3/8	2/8

Table 3: Set of CARs and their supports

R	**R_9**	**R_8**	**R_7**	**R_6**	**R_5**	**R_4**	**R_3**	**R_2**	**R_1**
Sup	2/8	3/8	2/8	3/8	2/8	2/8	2/8	2/8	2/8

Table 4: Set of CARs and their supports after first partition

R	**R_6**	**R_8**	**R_7**	**R_9**	**R_5**	**R_4**	**R_3**	**R_2**	**R_1**
Sup	3/8	3/8	2/8	2/8	2/8	2/8	2/8	2/8	2/8

Table 5: Set of CARs and their supports after second partition

3.4 Complexity Analysis

For the best case, the proposed algorithm has complexity $O(n)$, where n is the number of rules. In this case, the whole rule set is partitioned into two groups, with the first group having k rules. However, in the worst case, the rule set is partitioned into two groups where one group has one rule and the other group has $n-1$ rules. In this case, we need k loops to get the result, and thus the complexity is $O(n*k)$. When k is large, this case requires a lot of time to query the top-k rules. To avoid the worst case, we sort nodes in the first level of the MECR-tree in increasing order according to their supports and choose the support of middle node as x (refer to line 8 in Algorithm 1).

Table 6 compares the proposed algorithm with QuickSort- and InsertionSort-based algorithms.

No	**Case**	**QuickSort**	**QuickSort-based**	**InsertionSort-based**
1	Best case	$O(n*\log n)$	$O(n)$	$O(n*k)$
2	Worst case	$O(n^2)$	$O(n*k)$	$O(n*k)$
3	Average case	$O(n*\log n)$	$O(n*\log n)$	$O(n*k)$
4	Optimization		Using order of nodes in MECR-tree	

Table 6: Complexity of algorithms

4 Experiments

4.1 Datasets and Testing Environment

The algorithms used in the experiments were coded using C# 2012, and run on a laptop with Windows 8.1 OS, an i5-4200U 1.60-GHz CPU, and 4 GB of RAM.

Experimental datasets were downloaded from the UCI Machine Learning Repository (http://mlearn.ics.uci.edu). Their charesteristics are shown in Table 7.

Dataset	# of attributes	# of classes	# of distinct items	# of records
Breast	12	2	737	699
German	21	2	1077	1000
Lymph	18	4	63	148
Zoo	17	7	43	101
Vehicle	19	4	1434	846

Table 7: Characteristics of experimental datasets

Table 8 shows the number of CARs obtained from the experimental datasets with a given *minSup* for each dataset. *minConf* was set to 50%.

Dataset	*minSup* (%)	# of rules
Breast	0.3	13,870
German	5	19,343
Lymph	8	30,911
Zoo	10	116,813
Vehicle	0.2	126,221

Table 8: Number of CARs in each dataset

4.2 Mining Time

Figures 3-7 compare the mining time of the proposed algorithm with those of an InsertionSort-based algorithm [Mai, 13a] with various *k* values.

These figures show that the QuickSort-based algorithm is more efficient than the InsertionSort-based method in all cases. For example, considering the Lymph dataset, the runtimes for mining top-*k* rules using the InsertionSort-based method are 1.558, 4.948, 10.193, 17.193, 25.553 s for k = 2000, 4000, 6000, 8000, and 10,000, respectively. For the QuickSort-based algorithm, the mining times are 0.367, 0.38, 0.394, 1.357, and 0.411 s for k = 2000, 4000, 6000 and 8000, and 10,000, respectively. Similar results were obtained for the other datasets.

The experimental results show that the QuickSort-based algorithm is faster than the InsertionSort-based one. The proposed algorithm is efficient when datasets have many rules and the threshold k is very large. For example, with the German dataset and k = 10,000, the runtime for the QuickSort-based algorithm is 0.03 s and that for the InsertionSort-based one is 10.03 s.

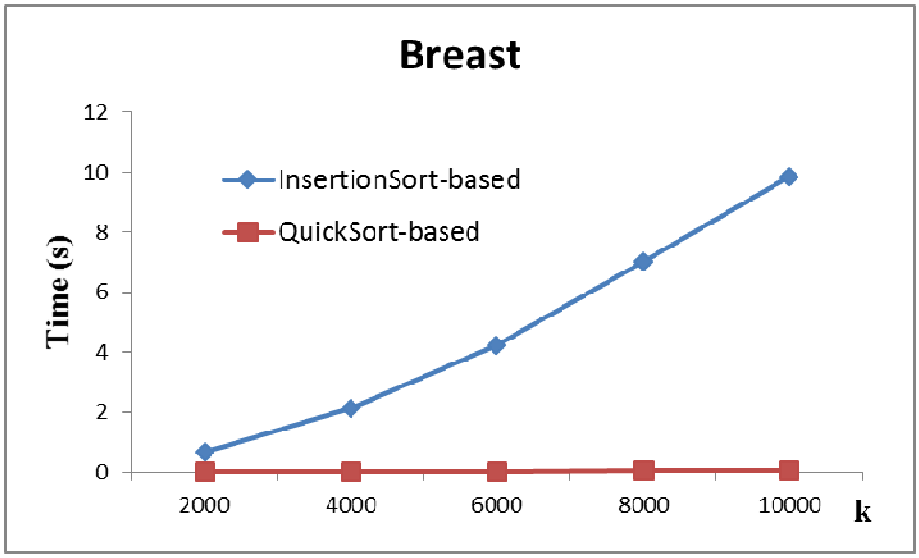

Figure 3: Mining times of InsertionSort-based and QuickSort-based methods for Breast dataset

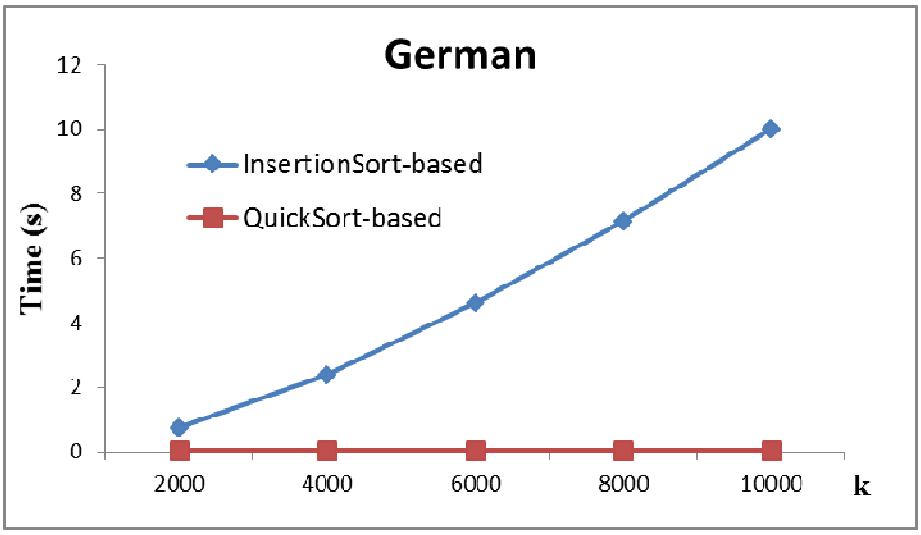

Figure 4: Mining times of InsertionSort-based and QuickSort-based methods for German dataset

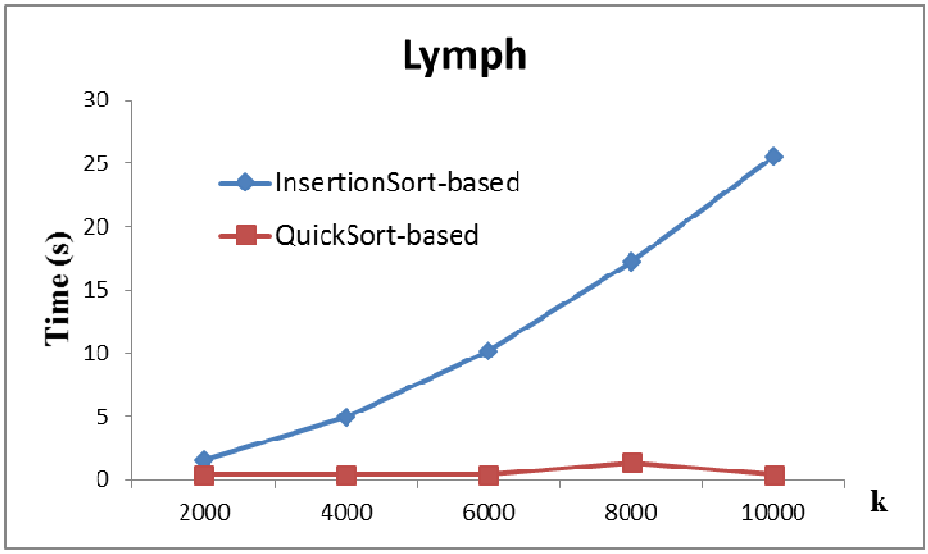

Figure 5: Mining times of InsertionSort-based and QuickSort-based methods for Lymph dataset

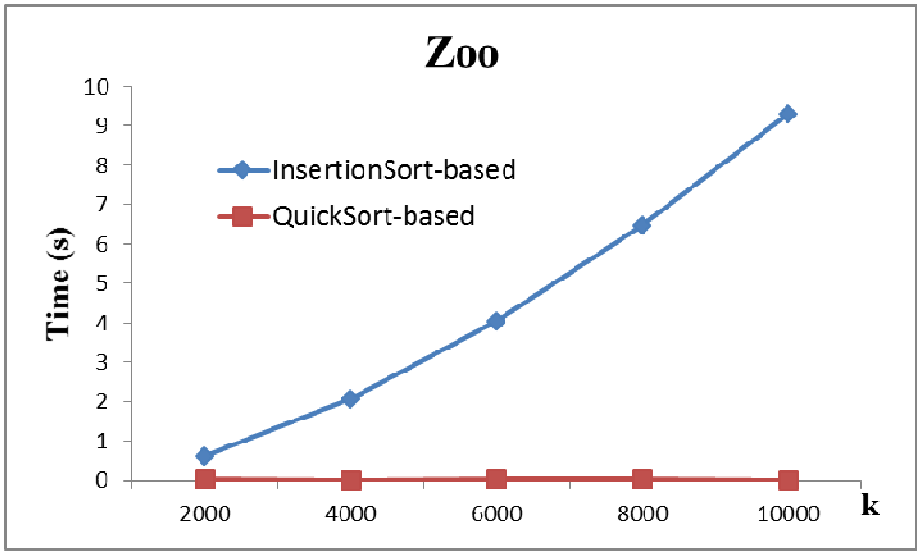

Figure 6: Mining times of InsertionSort-based and QuickSort-based methods for Zoo dataset

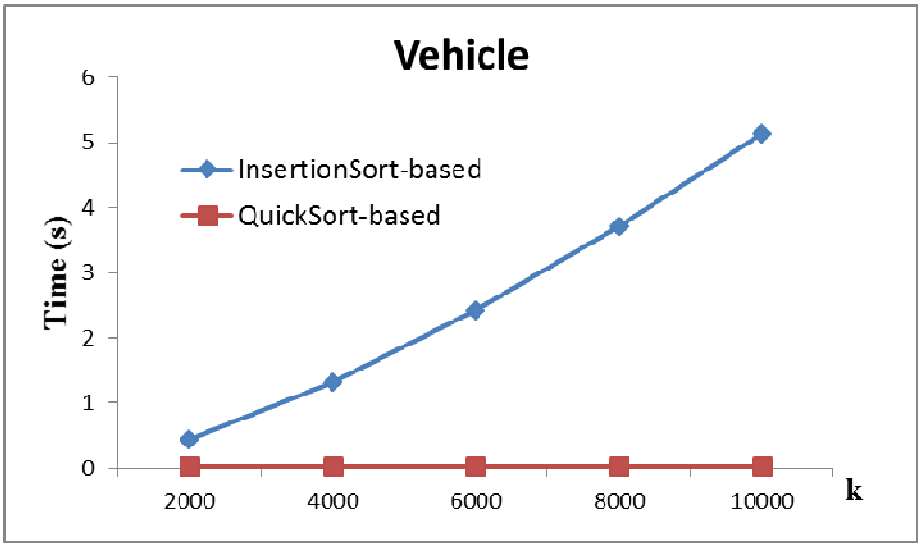

Figure 7: Mining times of InsertionSort-based and QuickSort-based methods for Vehicle dataset

5 Conclusion and Future Work

This paper proposed a method for mining top-*k* CARs using a QuickSort-based algorithm. In our algorithm, a divide-and-conquer scheme is used to significantly enhance performance. Some additional heuristics are used to avoid the worst case for the algorithm. The proposed algorithm is faster than InsertionSort-based method.

In the future, we will continue to study how to prune rules that cannot belong to top-*k* rules to reduce the search space. We will also extend our method to the mining of top-*k* non-redundant CARs and sequential rules [Van, 14]. For very large datasets, we are also interested in "anytime" algorithms [Mai, 13a; Mai, 13b; Mai, 15] to further enhance performance and provide results under some resource constraints.

Acknowledgements

This research is funded by NTTU Foundation for Science and Technology Development under grant number 2016.01.42.

References

[Breiman, 84] Breiman L., Friedman J.H., Olshen R.A., Stone C.J.: Classification and Regression Trees, In Proc. of Wadsworth, Belmont, CA. Republished by CRC, vol.1, 14-23, 1984.

[Chen, 12] Chen W.C., Hsu C.C., Hsu J.N.: Adjusting and generalizing CBA algorithm to handling class imbalance, Expert Systems with Applications, vol.39, no.5, 5907-5919, 2012.

[Chen, 14a] Chen F., Wang Y., Li M., Wu H., Tian J.: A Principal Association Mining An efficient classification approach, Knowledge-Based Systems vol.67, 16-25, 2014.

[Chen, 14b] Chen C.H., Chiang R.D., Lee C.M., Chen C.Y.: Improving the performance of association classifiers by rule prioritization, Knowledge-Based Systems, vol.36, 59-67, 2014.

[Deng, 07] Deng Z.H., Fang G.: Mining top-rank-k frequent patterns, ICMLC, 851-856, 2007.

[Deng, 14] Deng Z.H.: Fast mining top-rank-k frequent patterns by using Node-lists, Expert Systems with Applications, vol.41, no.4, 1763-1768, 2014.

[Do, 15] Do T.N., Lenca P., Lallich S.: Classifying many-class high-dimensional fingerprint datasets using random froest of oblique decision trees, Vietnam Journal Computer Science, vol.2, 3-12, 2015.

[Fournier-Viger,, 11] Fournier-Viger P., Tseng V.S.: Mining Top-K Sequential Rules, 7th International conference. In Proc. of ADMA 2011, Beijing, China, vol.7121, 180-194, 2011.

[Fournier-Viger,, 12a] Fournier-Viger P., Wu C.W., Tseng V.S.: Mining Top-K Association Rules, In Proc. of Canadian Conference on AI 2012, Toronto, Canada, vol.7310, 61-73, 2012.

[Fournier-Viger,, 12b] Fournier-Viger P., Vincent S. Tseng: Mining Top-K Non-redundant Association Rules, In Proc. of 20th International Symposium, ISMIS 2012, Macau, China, vol.7661, p. 31-40, 2012.

[Geoffrey, 11] Geoffrey I. W.: Filtered-top-k association discovery, WIREs Data Mining and Knowledge Discovery, vol. 1, 183-192, 2011.

[HooshSadat, 12] HooshSadat M., Zaïane O.R.: An associative classifier for uncertain datasets, In Proc. of PAKDD 2012, 342-353, 2012.

[Lan, 06] Lan Y., Janssens D., Chen G., Wets G.: Improving associative classification by incorporating novel interestingness measures, Expert Systems with Applications, vol.31, no.1, 84-192, 2006.

[Le, 15] Le Q.H.T., Le T., Vo B., Le B.: An efficient and effective algorithm for mining top-rank-k frequent patterns, Expert Systems with Applications, vol.42, no.1, 156-164, 2015.

[Li, 01] Li W., Han J., Pei J.: CMAR: Accurate and efficient classification based on multiple class-association rules, In Proc. of The 1st IEEE International Conference on Data Mining, San Jose, California, USA, 369-376, 2001.

[Liu, 98] Liu B., Hsu W., Ma Y.: Integrating classification and association rule mining, In Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, New York, USA, 80-86, 1998.

[Liu, 09] Liu H., Sun J., Zhang H.: Post-processing of associative classification rules using closed sets, Expert Systems with Applications, vol.36, no.3, 6659-6667, 2009.

[Nguyen, 13] Nguyen L.T.T., Vo B., Hong T.P., Thanh H.C.: CAR-Miner: An efficient algorithm for mining class-association rules," Expert Systems with Applications, vol.40, no.6, 2305-2311, 2013.

[Nguyen, 15a] Nguyen L.T.T., Nguyen N.T.: An improved algorithm for mining class association rules using the difference of Obidsets, Expert Systems with Applications. vol. 42, no. 9, 4361-4369, 2015.

[Nguyen, 15b] Nguyen L.T.T., Vo B., Hong T.P.: CARIM: An Efficient Algorithm for Mining Class-association Rules with Interestingness Measures, The International Arab Journal of Information Technology, vol.12, no.7, 627-634, 2015.

[Nguyen, 16] Nguyen L.T.T., Nguyen H., N.T., Vo B., Nguyen N.T.: An efficient method for query top-k rules from class association rule set, 8th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Da Nang, Vietnam, 2016.

[Mai, 13a] Mai S.T., He X., Feng J., Böhm C.: Efficient anytime density-based clustering, In Proc. Of SIAM International Conference on Data Mining (SDM'13), 112-120, 2013.

[Mai, 13b] Mai S.T., He X., Hubig N., Plant C., Böhm C.: Active Density-based Clustering, in Proc. of International Conference on Data Mining (ICDM'13), 508-517, 2013.

[Mai, 15] Mai S.T., He X., Feng J., Plant C., Böhm C.: Anytime density-based clustering of Complex Data, *Knowledge and Information System*, vol. 45, 319-355, 2015.

[Parker, 11] Parker C.: An Analysis of Performance Measures For Binary Classifiers, In Proc. of ICDM 2011, 517-526, 2011.

[Quinlan, 86] Quinlan J. R.: Introduction of decision tree, Machine Learning, vol.1, no.1, 81-106, 1986.

[Quinlan, 92] Quinlan J. R.: C4.5: program for machine learning, Morgan Kaufmann, 1992.

[Shaharanee, 11] Shaharanee I.N.M., Hadzic F., Dillon T.S.: Interestingness measures for association rules based on statistical validity, Knowledge-Based Systems, vol.24, 386-392, 2011.

[Thabtah, 04] Thabtah F., Cowling P., Peng Y.: MMAC: A new multi-class, multi-label associative classification approach, In Proc. of the 4th IEEE International Conference on Data Mining, Brighton, UK, 217-224, 2004.

[Thabtah, 05] Thabtah F., Cowling P., Peng Y.: MCAR: Multi-class classification based on association rule, In Proc. of The 3rd ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, 33-39, 2005.

[Thabtah, 07a] Thabtah F.A.: A review of associative classification mining, Knowledge Engineering Review, vol.22, no.1, 37-65, 2007.

[Thabtah, 07b] Thabtah F.A., Cowling P.I.: A greedy classification algorithm based on association rule, Applied Soft Computing, vol.7, no.3, p. 1102-1111, 2007.

[Tolun, 98a] Tolun M.R., Abu-Soud S.M.: ILA: An inductive learning algorithm for production rule discovery, Expert Systems with Applications, vol.14, no.3, 361-370, 1998.

[Tolun, 98b] Tolun M.R., Sever H., Uludag M., Abu-Soud S.M.: ILA-2: An inductive learning algorithm for knowledge discovery, Cybernetics and Systems, vol.30, no.7, 609-628, 1998.

[Van, 14] Van T.T., Vo B., Le B.: IMSR_PreTree: An improved algorithm for mining sequential rules based on the prefix-tree, Vietnam Journal Computer Science, vol.1, no.2, 97-105, 2014.

[Vo, 08] Vo B., Le B.: A novel classification algorithm based on association rule mining, In Proc. of the 2008 Pacific Rim Knowledge Acquisition Workshop (Held with PRICAI'08), LNAI 5465, vol.5465, 61-75, 2008.

[Vo, 11] Vo B., Le B.: Interestingness measures for association rules: Combination between lattice and hash tables, Expert Systems with Applications, vol.38, no.9, 11630-11640, 2011.

[Yin, 03] Yin X., Han J.: CPAR: Classification based on predictive association rules, In Proc. of SIAM International Conference on Data Mining (SDM'03), San Francisco, CA, USA, 331-335, 2003.