Context-Aware Role Mining for Mobile Service Recommendation

Jian Wang, Cheng Zeng, Chuan He, Liang Hong, Liang Zhou
State Key Lab of Software Engineering, Wuhan University, China
{jianwang,zengc,brook,hong,zhouliang979}@whu.edu.cn

Raymond K. Wong
School of Computer Science and Engineering, University of New South Wales, Australia
wong@cse.unsw.edu.au

Jilei Tian
Nokia Research Center, Beijing, China
jilei.tian@nokia.com

ABSTRACT

Finding and recommending suitable services for mobile users are increasingly important due to the popularity of mobile Internet. While recent research has attempted to use role-based approaches to recommend services, role discovery is still an ongoing research topic. In particular, numerous role mining approaches have been proposed in the context of RBAC (Role-Based Access Control) in which only two dimensions of parameters are considered (i.e., <user, permission>); and the notion of context-awareness has been disregarded. This paper proposes a context-aware role mining method to automatically group users according to their interests and habits, such that popular mobile services can be recommended to other members in the same group in a context dependent manner. Experiments show that our approach is efficient and practical for mobile service recommendation.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval –*Clustering.*

General Terms
Algorithms.

Keywords
Role Mining, Context, FCA.

1. INTRODUCTION

The popularity of smart phones poses many challenges. One of these challenges is to make these devices more intelligent and adaptive to user requirements by automatically recommending suitable services. This is especially important due to the increasing number of service offerings available on the Internet and mobile network, as a result of the emergence of Service Oriented Architecture (SOA) and cloud computing. For example, if a user always *listens to popular music between 8:00-9:00AM, while the user is on the bus on the way to office every weekday morning*, a pop music service can be automatically recommended to the user once such a context for the user is detected.

Current service recommendation techniques are mainly based on individual intelligence or the local knowledge of users [17], and do not take into consideration the common knowledge among different users. For example, referring to the previous example, a popular service to a group could have been recommended to all members of the group automatically. In this paper, we use the notion of "roles" to represent user groups. It is used to represent the abstract characterization of user behaviors within a certain context. Users can play different roles in their daily life, and their roles change dynamically as the context changes.

The use of roles has several benefits. Firstly, users who play the same role are likely to share the same preferences and behavior patterns. Once a user's role in a certain context is recognized, the services closely related to the role can be recommended to the user. For example, a user might be more willing to deal with incoming sharing requests during a shopping trip than during cooking time. If the system can know the role of the user is a housewife, such a request will not be sent during the cooking time. Secondly, the service suggestion from the person who plays the same role with the user is usually more reliable than those who seldom play the role. Furthermore, as pointed out by previous work on role modeling (e.g., [19]), roles can have different granularity or levels and are better represented in a hierarchy, e.g., in form of ontology.

Although the concept of roles has been used in task/service recommendation (e.g., [3]), most of the existing approaches typically implement the role-task ontology manually. This is infeasible in practice for mobile environments where both roles and contexts are changing dynamically.

This paper adapts the concept of roles from the domain of RBAC (Role-Based Access Control) [4] and organizes user interests and habits according to roles. We then recommend services according to the usages/experiences of the other users sharing the same roles. Existing role mining approaches in the field of RBAC [9] only consider two dimension parameters, namely <*user, permission*>, and neglect the "context", which is an important parameter in mobile service recommendation. In this paper, we extend the behavior pattern recognition method [5] (from user operation history) to identify context-aware roles from multi-user behavior patterns. We then organize these roles into a hierarchical structure in form of ontology.

Overall, our proposed method consists of the following main steps: (1) Mine the minimal set of roles from the table of (user, context, behavior), and output a user-role table and a role-context-behavior table; and (2) Create a role tree based on the FCA (Formal Concept Analysis) method, which is then stored in a RDF (Resource Description Framework) file. Based on the generated role ontology, services shared according to roles will be recommended to the corresponding users.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the context-aware role mining problem and Section 4 presents our proposed algorithm. Experiments running on a mobile device are presented in Section 5 and Section 6 concludes the paper.

2. RELATED WORK

The concept of roles has been used in many areas such as linguistics, knowledge representation, relational database, and access control [3]. In [6], the definition of roles was given as: 1) role is a property assigned to humans, and roles can change dynamically, and 2) humans can have multiple roles simultaneously. Roles have been widely used in data security domain, under RBAC [4]. [8] proposed an effective ontology development system called Hozo for organizing role-concepts. In [3], role was mentioned as a definitive factor in task or service selection. Two kinds of roles were presented: social role and task role. In their paper, role ontology needs to be manually created to determine task candidates and recommend services. However, since the role ontology is manually created, their technique does not scale for many users or tasks.

In the field of RBAC, numerous role mining approaches have been proposed [9]. For example, [10] proposed a hybrid role mining methods that work with both direct user-permission assignments and business information from the enterprise. They used statistical measures to analyze the relevance of different kinds of business information for defining roles. In [11], the role mining problem (RMP) was defined as the problem of discovering an optimal set of roles from existing user permissions. The paper formally defined RMP and analyzed its theoretical bounds. By relating RMP with the database tiling problem in data mining domain, an effective RMP algorithm was proposed.

While context is an important factor for mobile service recommendation, it has been mostly disregarded in RBAC. To the best of our knowledge, [12, 20] are the only papers addressing context-aware RBAC. A context-aware RBAC (CARBAC) model for pervasive computing applications was presented in [12], in which the context management layer and the access control layer were distinguished. The model supported personalized permissions for role members, and context-based constraint specifications. In [20], a complete RBAC model in spatiotemporal domain based on the idea of spatiotemporal context was proposed. The concept of spatiotemporal role extent and spatiotemporal permission extent introduced there could model granular spatiotemporal access control policies not specifiable in the existing approaches, which would be particularly important for mobile computing [20].

However, both [12] and [20] did not describe how to discover or identify context–aware roles. Traditional RBAC-based role mining algorithms (such as [10] or [11]) cannot be directly applied, since traditional RBAC models only consider two-dimensional parameters, namely <*user, permission*>, while context-aware role mining approach needs to consider at least three-dimensional dataset <*user, context (time+scene), behavior*> (here we consider *context* as *time* and *scene,* which will be explained later in Section 3).

3. THE CONTEXT-AWARE ROLE MINING PROBLEM

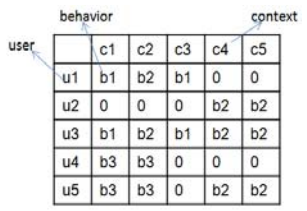
In general, roles can be classified as *continuous roles* and *periodic roles* according to the steady and temporal features that a user plays, respectively. Periodic roles can be mined from the behavior patterns of multiple users, and continuous roles can be mined from the static information in user profile and also from the periodic roles. For example, if a group of users with similar age frequently plays the same periodic roles, all of them possibly play some common continuous roles.

A role can be denoted as a set of preference triple of the format <*time, scene, behavior*>. For instance, consider a preference triple <*business hours, on bus, listen to music*>, tagged with a periodic role *music lover*.

The context-aware role mining problem in this paper is defined as follows.

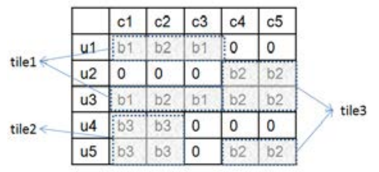
Definition. *Given a user behavior pattern configuration bpc =<U, C, B, UCB>, where U is a set of users, C is a set of context, B is a set of behaviors,* $UCB \subseteq U \times C \times (B \cup \{0\})$ *is the user-context-behavior relation, RMP is to find a state <R, UA, CBA> that is consistent with bpc, such that for any <R', UA', CBA'> that is consistent with bpc, #R ≤ #R', where R, R' is a set of roles,* $UA, UA' \subseteq U \times R$ *is the user-role assignment matrix,* $CBA, CBA' \subseteq R \times C \times (B \cup \{0\})$ *is the role-context-behavior assignment matrix. A state is consistent with bpc, if every user in U has the same set of context-behavior assignment as in bpc.*

Table 1. An example of role mining


behavior
context
user

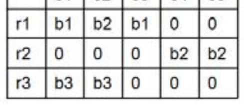
	c1	c2	c3	c4	c5
u1	b1	b2	b1	0	0
u2	0	0	0	b2	b2
u3	b1	b2	b1	b2	b2
u4	b3	b3	0	0	0
u5	b3	b3	0	b2	b2

(a)The original user-context-behavior matrix

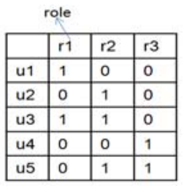

tile1
tile2
tile3

	c1	c2	c3	c4	c5
u1	b1	b2	b1	0	0
u2	0	0	0	b2	b2
u3	b1	b2	b1	b2	b2
u4	b3	b3	0	0	0
u5	b3	b3	0	b2	b2

(b)The selected tiles



	c1	c2	c3	c4	c5
r1	b1	b2	b1	0	0
r2	0	0	0	b2	b2
r3	b3	b3	0	0	0



	r1	r2	r3
u1	1	0	0
u2	0	1	0
u3	1	1	0
u4	0	0	1
u5	0	1	1

(c) The role-context-behavior matrix (d) The user-role matrix

For example, suppose the user set *U*={*u1,u2,u3,u4,u5*}, the context set *C*={*c1,c2,c3,c4,c5*}, and the behavior set *B*={*b1,b2,b3*}. The *UCB* can be shown in the user-context-behavior matrix, which is constructed according to users' behavior patterns, in Table 1(a). In Table 1(a), each column denotes a context, each row denotes a user, and the value *b* in cell {*i*, *j*} represents that the user *ui* has the behavior *b* under the context *cj*. Please note that *b*=0 means that the user does not have any behavior under that context.

After mining roles from the behavior pattern configuration, we obtain a state <*R, UA, CBA*>, where *R*={*r1*, *r2*, *r3*}, *UA* and *CBA* are shown as in Table 1(c) and 1(d), respectively. As shown in Table 1(c), three mined roles *r1*, *r2*, and *r3* are listed in a role-context-behavior matrix, e.g., *r1* has behavior *b1* under context *c1* and *c3*, and behavior *b2* under context *c2*. Table 1(d) shows a user-role matrix, where a 1 in cell {*i, j*} represents that the user *ui* can play the role *rj*. e.g., user *r1* can play role *r2*.

To mine the roles above, we extend the role mining method proposed in [13]. In [13], the theory of tiling database is used in role mining for RBAC applications. However, different from 0/1 in tiling databases and RBAC, the value in each cell of the user-context-behavior matrix is non-binary. So the definition of the tile is modified as a region in the user-context-behavior matrix where all elements are non-zero and the elements in each row are the same. The number of cells is named as the area of the tile. For example, in Table 1(b), the cells {(1,1), (1,2), (1,3), (3,1), (3,2),(3,3)} constitute a tile with the area 6.

Obviously, a large tile set consisting of only a small number of tiles can be extremely informative. The role mining problem can be reduced to the minimum tiling problem - which asks for a tiling of which the area equals to the total number of non-zero cells in the matrix and consists of the minimum number of tiles. Each tile corresponds to a potential role. Table 1(b) shows a minimal tiling of the matrix that consists of 3 tiles, as shown in the shaded region, i.e., Tile 1={(1,1), (1,2), (1,3), (3,1), (3,2), (3,3)}, Tile 2={(2,4), (2,5), (3,4), (3,5), (5,4), (5,5)} and Tile 3={(4,1), (4,2), (5,1), (5,2)}. It is impossible to find a tiling that covers the entire matrix with less than 3 tiles. These three tiles correspond to the three potential roles in Table 1(c).

Next, we employ the FCA (Formal Concept Analysis) approach [14] to create a role tree to hold all the mined roles. FCA is formalism for knowledge representation based on the formalization of "concepts" and "concept hierarchies". Its structures can be graphically represented as concept lattices, which allow the analysis of complex structures and the discovery of dependencies within the data. Moreover, concept lattices are easy for people to read and understand.

In FCA, a formal context can be represented as a matrix, where rows represent objects, and columns represent attributes. Cell (row *g*, column *m*) means that the object *g* has the attribute *m*. In order to use the FCA method, we need to first transform the role-context-behavior matrix into the formal context. Then, a role lattice can be generated based on the classic FCA algorithm.

4. THE ROLE MINING ALGORITHM

In this section, we present our proposed context-aware role mining method. The basic steps of this method are as follows: (1) Mine the minimal set of roles from the user-context-behavior matrix, and output the user-role assignment matrix and the role-context-behavior assignment matrix; and (2) Create a role tree based on FCA method and store it in RDF.

4.1 Mine the minimal role set

The work [13] proposes a depth first search algorithm to find the minimum tiling of any given database, which can be approximated within the factor *O(log mn)*. The work [11] proves that the minimum tiling problem is equivalent to the basic role mining problems in RBAC. However, this method cannot be directly applied in context-aware role mining. It is because that the methods in the tiling database and RBAC only consider the dataset in two dimensions, while the dataset to be considered in the context-aware role mining is three dimensional, i.e., user, context and behavior. Therefore, we will need to extend the algorithm as follows.

```
Algorithm 1: Mine the Minimal Role Set
1: Matrix ← Read the user-context-behavior matrix;
2: Tiles ← MineTileSet(Matrix);
3: for each tile t∈Tiles do
4:    create a new role r;
5:    extract the set of context C and behaviors B in the tile;
6:    for each context c∈C, behavior b∈B do
7:       add the assignment {r,c,b} to role-context-behavior matrix;
8:    end for
9:    extract the set of users U in the tile;
10:   for each user u ∈U do
11:      add the assignment {u, r} to user-role matrix;
12:   end for
13: end for
```

In Algorithm 1, Line 1 reads the user-context-behavior matrix. Line 2 finds a minimum tile set for the given matrix, and the detail of the algorithm will be given in Algorithm 2 below. Based on the tiles mined, Lines 3-13 create new roles, update the role-context-behavior assignment matrix and the user-role assignment matrix.

```
Algorithm 2: MineTileSet(Matrix)
1: Tiles←{};
2: extract the set of users U, context C and behaviors B in Matrix;
3: for each context ci∈C in Matrix do
4:    subtile[totalbehavior]←{};
5:    initial[totalbehavior]←{};
6:    for each user u∈U in ci do
7:       add u to initial[Matrix[u][ci]];
8:    end for
9:    conditionaldb←CreateConditionaldb(ci,initial);
10:   contextI←{};
11:   add ci to contextI;
12:   ComputeTile(contextI,conditionaldb,subtile);
13:   for all i<totalbehavior do
14:      if subtile[i].tilearea> MINU×MINC
15:         add subtile[i] to Tiles;
16:      end if
17:   end for
18: end for
19: RemoveDuplication(Tiles);
20: Return Tiles.
```

In Algorithm 2, we need to first determine which kind of tiles can become a candidate role. Without loss of generality, the minimal requirement to form a role is assumed to have at least two users sharing the same behaviors under two different contexts, so a qualified candidate tile will cover at least two users and two contexts. We define two parameters MINU (≥2) and MINC (≥2) to denote the minimal requirements of the number of users and contexts, respectively, to form a role. Lines 1-2 initialize the tile set, and extract the user set, context set, and behavior set from the given matrix. Note that each column in the matrix denotes a context. In lines 3-18, for each context in the matrix, the largest tile for each behavior occurring in this context can be computed, and the tile should start from this column. More specifically, in line 4, the *subtile* defines an array to store the tiles with the largest area for each behavior. The variable *totalbehavior* is used to record the total number of the behaviors in the behavior set. Lines 5-8 are used to divide the users into different groups according to their values in column *ci*. For example, in Table 1, for the column *c1*, initial[b1]={u1,u3}, initial[b3]={u4,u5}. Line 9 is used to create the conditional database. The procedure to create conditional database is given in Algorithm 3. Lines 10-11 record the context that has been considered. Line 12 invokes Algorithm 4 recursively to find the largest uncovered tiles in the conditional database. Lines 13-18 delete the candidate tiles whose areas are less than MINU×MINC. Finally Line 19 removes the duplicated tiles from the tile set, and the details are given in Algorithm 5.

The basic idea of the tile mining algorithm is that all large tiles containing item *i*, but not containing any item smaller than *i*, can be found in the so called *i*-conditional database [13,15]. Since the dataset we faced with is three-dimensional, the definition of the conditional database is slightly different. In our context, the *i*-conditional database is defined as the union of intersections of the column *i* and the columns that are bigger than *i*. For example, in the example of Table 1(a), the *c1*-conditional database={*cdb*(*c1*∩*c2*), *cdb*(*c1*∩*c3*), *cdb*(*c1*∩*c4*), *cdb*(*c1*∩*c5*)}, where *cdb*(*c1*∩*ci*) denotes the intersection of column *c1* and *ci*. We also classify the cells in *cdb*(*c1*∩*ci*) as *subcdb* according to their values. As shown in Figure 1, which is based on Table 1(a), *cdb*(*c1*∩*c2*)={{(*u1,c1*), (*u1,c2*), (*u3,c1*), (*u3,c2*)}, {(*u4,c1*), (*u5,c2*), (*u4,c1*), (*u5,c2*)}}, two *subcdbs* can be classified according to the value of the cell is *b1* or *b3*. *subcdb1*={*(u1,c1*), (*u1,c2*), (*u3,c1*), (*u3,c2*)}, *subcdb2*={(*u4,c1*), (*u5,c2*), (*u4,c1*), (*u5,c2*)}. In fact, the *subcdb* can be viewed as a potential tile, which can correspond to a candidate role. Once the algorithm is iterated, more conditional databases can be computed. As shown in the below part of Figure 1, the *c1&c2*-conditional database ={{*(u1,c1*), (*u1,c2*), (*u1,c3*), (*u3,c1*), (*u3,c2*), (*u3,c3*)}}, which consists of a *subcdb subcdb3*={(*u1,c1*), (*u1,c2*), (*u1,c3*), (*u3,c1*), (*u3,c2*), (*u3,c3*)}. Please note that *subcdb1* is a subset of *subcdb3*, and we cannot find any *subcdb* that is larger than *subcdb3*, so *subcdb3* is a tile that can be returned. Similarly, *subcdb2* is also a tile. These tiles can become candidate roles. This is essentially how our role mining algorithm works.

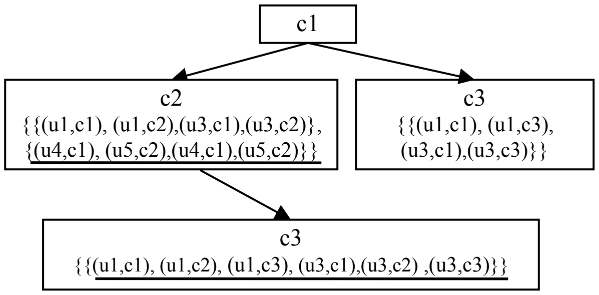

c1
c2
{{(u1,c1), (u1,c2),(u3,c1),(u3,c2)},
{(u4,c1), (u5,c2),(u4,c1),(u5,c2)}}
c3
{{(u1,c1), (u1,c3),
(u3,c1),(u3,c3)}}
c3
{{(u1,c1), (u1,c2), (u1,c3), (u3,c1),(u3,c2) ,(u3,c3)}}

Figure 1. An example of a conditional database

Algorithm 3 introduces how to create a *c*-conditional database in detail. Line 1 initializes the conditional database. As shown in Line 2, each column that is bigger than *c* will be considered. Line 3 initializes the *cdb* that stores the intersection of *c* and *cj*. Lines 4-17 show how to compute the *cdb*. Since each *cdb* can be split into different *subcdbs* according to the different behaviors in the cells, all the possible behaviors are considered in line 4. Since we are creating a *c*-conditional database, the users have already been classified according to their behaviors under context *c*, which is recorded in cUser. If there is at most one user having a certain behavior under context *c*, it is impossible to find a tile according to this user. Line 5 can ignore such a case. Lines 6-10 create a *subcdb* to guarantee that the users in *subcdb* have the same behaviors under context *c* and *cj*, respectively. Line 9 is used to record which behavior the users in *subcdb* have under context *c*, which is used in Algorithm 4. Lines 11-15 are used to check that each *subcdb* (that can be added to *cdb*) corresponds to at least MINU users. For example, in Table 1(a), *cdb*(*c1*∩*c4*)= {{(*u3,c1*), (*u3,c4*)}, {(*u5,c1*), (*u5,c4*)}}, both of the two *subcdbs* {(*u3,c1*), (*u3,c4*)} and {*(u5,c1*), (*u5,c4*)} have only one user. So *cdb*(*c1*∩*c4*) will be ignored.

Algorithm 3: CreateConditionaldb(c, cUser)
```
1: conditionaldb←{};
2: for each cj∈C and cj>c do
3:    cdb←{};
4:    for each i<totalbehavior do
5:       if cUser[i].size≥ MINU
6:          subcdb[totalbehavior]←{};
7:          for each user u in cj do
8:             add u to subcdb[Matrix[u][cj]];
9:             subcdb[Matrix[u][cj]].ancestor=Matrix[u][c];
10:         end for
11:         for all i<totalbehavior do
12:            if subcdb[i].size≥ MINU
13:               add subcdb[i] to cdb;
14:            end if
15:         end for
16:      end if
17:   end for
18:   add cdb to conditionaldb;
18: end for
19: Return conditionaldb
```

Algorithm 4: ComputeTile(contextI,conditionaldb,mysub)
```
1: for each cdb ∈conditionaldb do
2:    for each subcdb ∈cdb do
3:       if (contextI.size+column- subcdb.context)* subcdb.user.size <
         mysub[subcdb.ancestor].area
4:          remove subcdb from cdb;
5:       end if
6:          if subcdb.user.size*(contextI.size+1)> mysub [subcdb.
            ancestor].area
7:          mysub[subcdb.ancestor]=subcdb;
8:       end if
9:    end for
10: end for
11: for each cdb∈conditionaldb do
12:   newConditionaldb←CreateConditionaldb(cdb.context, cdb. user);
13:    add cdb.context to contextI;
14:   ComputTile(contextI,newconditionaldb,subtile);
15:  end for
```

Algorithm 4 is used to compute the largest uncovered tiles in the given conditional database. Lines 3-5 prune the *subcdbs* that are impossible to have larger area than the *subcdb* found in the former column. Lines 6-8 traverse the conditionaldb to find the largest *subcdb* in each *cdb*. Line 12 creates a new conditional database based on the current *cdb*. For example, if the current *cdb* in Figure 1 is *cdb*(*c1*∩*c2*) that is a element of *c1*-conditional database, then the new conditional database is *c1&c2*-conditional database. Line 13 records the context that *cdb* corresponds to in the context. On line 14, the algorithm is called recursively to find all large tiles in the new conditional database.

```
Algorithm 5: RemoveDuplication(Tiles)
1: for each tile t1 in Tiles do
2:     for each tile t2 in Tiles do
3:         if all cells in t1 are in t2
4:             remove t1 from the Tiles;
5:         end if
6:     end for
7: end for
```

In Algorithm 5, the duplicated tiles will be removed from the tile set. We need to discuss what kinds of tiles should be deleted. In Table 2, the left table shows the original user-context-behavior matrix, two tiles can be found as shown in the shaded area. Two roles *r1* and *r2* can be created as shown in the right table. It is obvious that *r2* is a subset of *r1*, and thus an intuitive idea is to delete *r2*. However, *r2* should not be removed. Since two users *u1* and *u2* will play the role *r1*, and four users *u1,u2,u3* and *u4* will play the role *r2*, if *r2* is deleted, the common knowledge among the four users can not be recorded. As long as the tile of *r2* is not a subset of the tile of *r1, r2*, it should not be deleted. Therefore, we only delete the tiles that are subset of other existing tiles.

Table 2. An example of removing duplication

	c1	c2	c3
u1	b1	b2	b1
u2	b1	b2	b1
u3	b0	b2	b1
u4	b3	b2	b1

	c1	c2	c3
r1	b1	b2	b1
r2	0	b2	b1

4.2 Create the role tree

Based on the mined roles in a role-context-behavior matrix, we will show in Algorithm 6 how to mine a role tree based on the FCA. Since FCA has been widely investigated in many domains, several algorithms and software have been proposed, e.g., In-Close, an open source software, is a fast formal concept miner for FCA files [16]. We adopt the In-Close as the core algorithm to create the role concept lattice.

In Algorithm 6, we first need to convert the role-context-behavior matrix into a formal context in FCA. The formal context will be a two-dimensional matrix, where rows represent roles, and columns represent the pairs of context-behavior. On line 2, the In-Close algorithm will be invoked. The role concepts and their relationships can be mined. A concept lattice represented in PNG can be output. Based on the PNG figure, domain experts can determine which kinds of roles can correspond to roles in reality. Since the selected roles as well as their relationships will be incorporated into our user profile ontology represented in RDF, the role concept lattice are also represented in RDF.

```
Algorithm 6: CreateRoleTree(role-context-behavior matrix)
1: ConvertToCXT();
2: InCloseMin();
3: OutputPNG();
4: OutputRDF();
```

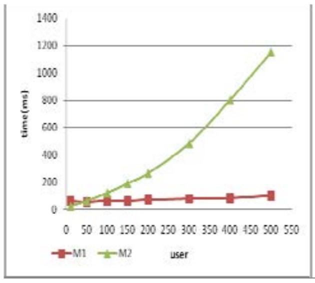
5. EXPERIMENTS AND DISCUSSION

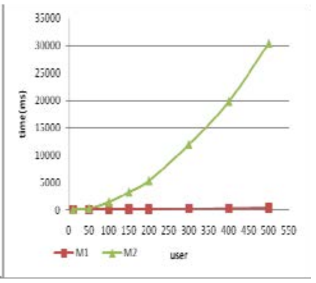
The context-aware role mining algorithm is implemented in C++ and run on Nokia N900, while the evaluation framework that hooks to the device is a PC with Intel Core 2 T7300, 2GHz CPU, 2GB main memory, and running Windows XP.

In this section, M1 represents our proposed method (presented in Section 4), and M2 is the classical FCA method.

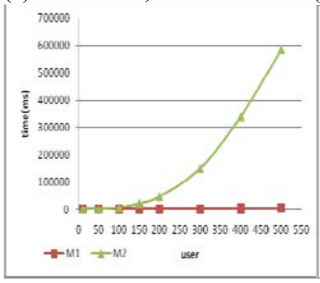
Artificial dataset: For the experiments using an artificial dataset, randomly generated user-context-behavior matrices are used. For each user and each context, a behavior can be randomly selected from the behavior set.

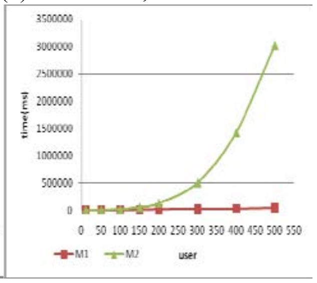
Figure 2 shows the comparison of the computation time (in ms) of the two methods M1 & M2 when the context and behavior sets are fixed. In Figure 2(a-d), the numbers of the context and behavior sets are set at 3, 5, 10, and 20, respectively. Each value in the figure is the average of 5 tests. From the results shown in the figure, we can conclude that M1 outperforms M2 significantly.





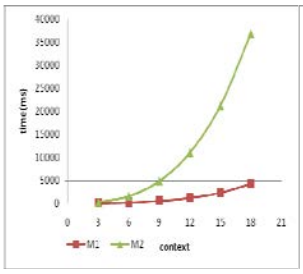
(a)#Context=3, #Behavior=3 (b)#Context=5, #Behavior=5

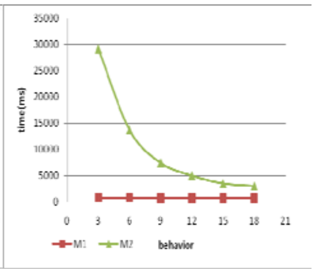




(c)#Context=10, #Behavior=10 (d)#Context=20, #Behavior=20

Figure 2. The comparison of mining time





(a)#User=100, #Behavior=10 (b)#User=100, #Context=10

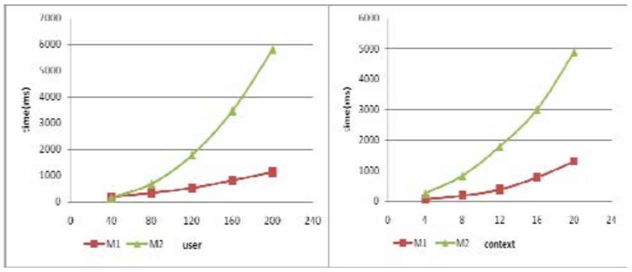
Figure 3. The mining time when users are fixed

Figure 3(a) shows the computation time (in ms) when the user set and behavior set are fixed. Figure 3(b) shows the computation time (again, in ms) when the user set and context set are fixed. We can see when #User=100 , and #Behavior=10, the computation time significantly increases when the context set increases. However, when #User=100, #Context=10, the computation time slightly decreases when the behavior set increases. This is because more contexts mean more columns in the matrix, and they will increase the number of iterations in the algorithm. However, the number of iterations does not increase when behavior number increases (in fact the area of each tile may decrease).

Real-life dataset: In addition to the randomly generated data set, we have also conducted two experiments (Figure 4) based on real-life data from users. We have collected the usage habits and behavior patterns of 240 mobile users based on an online questionnaire site (URL: http://202.114.107.230:8080/MY/). In the questionnaire, 16 behaviors such as listening to music, shopping online, watching movie, and reading news are pre-defined. The context is defined as the Cartesian product of Day, TimePeriod, and Scene, where Day={*weekday, weekend*}, TimePeriod={*morning, noon, afternoon, night*}, and Scene={*indoor, outdoor, on the bus*}, where *indoor* can be classified as *classroom*, *cinema*, *bedroom*, etc.; *outdoor* can be classified as *playground*, *bus*, *subway*, etc.

Based on the collected data set, as shown in Figure 4(a) and 4(b), M1 is more scalable than M2 when the number of users increases or the number of contexts increases. These results are consistent with the experiments based on randomly generated data.

We also note that the actual performance using real-life data is better than using randomly generated data, as there are generally more 0s in the real-life data matrices.



(a) #Context=16, #Behavior=16 (b) #User=200, #Behavior=16

Figure 4. The comparison of mining time on real-life data

To ensure the correctness of our mined results, we have compared the generated role-context-behavior matrix and the user-role matrix with the initial user-context-behavior matrix. We have also received feedbacks from the users to rate if the mined roles describe their habits and behavior patterns. The users have rated the results based on 1 or "Disappointed", 2 for "Not satisfied", 3 for "Ordinary", 4 for "Satisfied", and 5 for "Very Satisfied". The average score from all the 240 users is approximately 4.1.

6. CONCLUSIONS

By extending the research result from the field of RBAC, in this paper, we have proposed a context-aware role mining method to automatically group users according to their interests and habits, such that popular mobile services can be recommended to other members in the same group in a context dependent manner. Experiments based on randomly generated data and real-life data have shown that our proposed method is efficient and scalable for mobile service recommendation. For ongoing research, we are currently working with a handset manufacturer to evaluate the method with a large amount of user data and on a variety of devices.

7. ACKNOWLEDGMENTS

This work is supported by Nokia Research Center (Beijing), National Natural Science Foundation of China No.60970017, 61100018, the Fundamental Research Funds for the Central Universities No.3101034, and the Doctoral Fund of Ministry of Education of China No. 20100141120050.

8. REFERENCES

[1] Ghosh, R., and Dekhil, M. Discovering User Profiles. In *Proceedings of WWW 2009*, 2009, 1233-1234.

[2] Context-aware Recommender Systems in Mobile Scenarios, http://www11. in.tum.de/forschung/projekte/cars.

[3] Fukazawa, K. F. Y., Naganuma, T., and Kurakake, S. Construction and Use of Role-ontology for Task Based Service Navigation System. In *Proceedings of ISWC 2006*, 2006, 806-819.

[4] Sandhu, R. S., Coyne, E. J., Feinstein, H.L., and Youman, C. E. Role-based access control models. *IEEE Computer*, 29, 2(1996), 38–47.

[5] Cao, H., Bao, T., Yang, Q., Chen, E., and Tian, J. An Effective Approach for Mining Mobile User Habits. In *Proceedings of the 19th ACM international conference on information and knowledge management*, 2010.

[6] Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., and Guarino, N. Social roles and their descriptions. In *Proceedings of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning*, 2004, 267–277.

[7] Biddle, B. J. *Role Theory: Expectations, Identities, and Behaviors*. New York: Academic Press, 1979.

[8] Sunagawa, E., Kozaki, K., Kitamura, Y. and Mizoguchi, R. Organizing role-concepts in ontology development environment: Hozo, AI Technical Report, Artificial Intelligence Research Group, I.S.I.R., Osaka Univ., 2004, 453–468.

[9] Frank, M., Buhmann, J. M., and Basin, D. A. On the definition of role mining. In *Proceedings of SACMAT 2010*, 2010, 35-44.

[10] Frank, M., Streich, A. P., Basin, D. A., and Buhmann, J. M. A probabilistic approach to hybrid role mining. In *Proceedings of 2009 ACM Conference on Computer and Communications Security*, 2009, 101-111.

[11] Vaidya, J., Atluri, V., and Guo, Q., The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of SACMAT 2007*, 2007, 175-184.

[12] Kulkarni, D., and Tripathi, A. Context-aware role-based access control in pervasive computing systems. In *Proceedings of SACMAT 2008*, 2008, 113-122.

[13] Geerts, F., Goethals, B., and Mielikainen, T. Tiling databases, Discovery Science, Lecture Notes in Computer Science, 2004, 278 – 289.

[14] Priss, U. Formal Concept Analysis in Information Science, *Annual Review of Information Science and Technology*, 40(2006), 521-543.

[15] Han, J., Pei, J., Yin, Y., and Mao R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8, 1(2004), 53–87.

[16] In-close Project, http://inclose.sourceforge.net/.

[17] Su, X., and Khoshgoftaar, T. M. A Survey of Collaborative Filtering Techniques, *Advances in Artificial Intelligence*, 2009.

[18] Adomavicius, G., and Tuzhilin, A. Context-Aware Recommender Systems, Recommender Systems Handbook, 2011, 217-253.

[19] Wong, R. K., Chau, H.L., and Lochovsky, F.H. A Data Model and Semantics of Objects with Dynamic Roles. In Proceedings of ICDE 1997, 1997, 402-411.

[20] Aich, S., Modal, S., Sural, S., and Majumdar, A.K. Role Based Access Control with Spatiotemporal Context for Mobile Applications. In *Transactions on Computational Science* IV, Springer-Verlag, 2009.