Mining Emerging Patterns of PIU from Computer-Mediated Interaction Events

Yaxin Yu[1(✉)], Ke Yan[2], Xinhua Zhu[3], Guoren Wang[1], Dan Luo[3], and Suresh Sood[3]

[1] College of Information Science and Engineering, Northeastern University, Shenyang, China
{Yuyx,Wanggr}@mail.neu.edu.cn
[2] College of Software, Northeastern University, Shenyang, China
Yanke1992.yk@gmail.com
[3] QCIS, University of Technology, Sydney, Australia
{Xinhua.Zhu,Dan.Luo,Suresh.Sood}@uts.edu.au

Abstract. It has been almost 20 years since Internet services became an integral part of our lives. Especially recent popularization of SNS (Social Network Services) such as Facebook, more and more people are attracted to Internet. Internet provides many benefits to people, but yields a consequent disturbing phenomenon of obsession with Internet, which is called PIU (Pathological Internet Use) or IAD (Internet Addiction Disorder) in academia. PIU or IAD has negative effects on people's health of mind and body, therefore, it is necessary to detect PIU. Among tools of surfing Internet, since computer is the most widely interactive media, it is significant to mine PIU emerging patterns from human-computer interaction events. As a result, an emerging pattern mining method based on interactive event generators, called PIU-Miner, is proposed in this paper. Experimental results show that PIU-Miner is an efficient and effective approach to discovering PIU.

Keywords: Emerging pattern · PIU · Computer-mediated interaction · Complex event · Generator

1 Introduction

Since Internet has widely spread over the world, using computer to surfing Internet has been a basic part of our daily life. Especially, with the popularity of SNS in recent years, Internet users exploded in exponential scale. Unfortunately, some heavy users suffer from extreme dependency on Internet, which affects their work, study and living severely. This phenomenon is named as Internet Addiction Disorder (IAD) by Goldberg in 1996 [1] or Pathologica Internet Use (PIU) by Young [2]. A common approach to diagnosing PIU or IAD is based on diagnostic questionnaire made by medical or psychology specialists such as Young's 20 items questionnaire [3] and Beard's 5 criteria [4]. However, to our best knowledge, no

L. Cao et al. (Eds.): ADMI 2013, LNAI 8316, pp. 66–78, 2014.
DOI: 10.1007/978-3-642-55192-5_6,

existing work discusses how to diagnose PIU according to computer-mediated interaction information.

Aiming at the issues mentioned above, in this paper, we proposed an emerging pattern detecting model, named PIU-Miner, to detect PIU. Referring to the emerging patterns (EPs) mining idea introduced by Li *et al.* [6], PIU-Miner only mined minimum EPs of interaction events, i.e., Generators, to detect PIU. Instead of mining all emerging patterns of interactive events, the efficiency of PIU-Miner model is improved dramatically. The main contributions of this paper are summarized as follows.

1. An emerging pattern discovering model, i.e., PIU-Miner, is proposed to detect PIU by mining computer-mediated interaction events, which successfully solve PIU discovering issues from computer's data mining viewpoint other than from traditional medical and psychology perspective.
2. Instead of mining all EPs of events, only minimum EPs, i.e., generators are mined in PIU-Miner model. At the same time, a score criterion of ranking generators to reduce the number of generators is also exploited. All these methods improve the mining effectiveness of PIU-Miner greatly.
3. For dealing with excessive fragmentation of complex event's occurrence time and lasting time, Occurrence Time Mapping Strategy (OTMS) and Duration Rounding Strategy (DRS) are proposed in this paper. Based on these two strategies, multiple complex events with similar semantic meaning can be merged to a certain degree so as to reduce excessive numbers of frequent event itemsets.
4. Extend frequent itemset mining to frequent event itemset mining, which gives the practical experience worth referring to generalize definitions, properties and corollaries in transaction database.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries such as simple event, complex event, equivalence class and generator. The details of PIU-Miner model and generator mining algorithm of complex events are discussed in Sect. 3. In Sect. 4, experimental results and evaluation are described. Section 5 gives some related work, while Sect. 6 concludes the paper and suggests the future work.

2 Preliminaries

Before introducing the PIU-Miner model, some basic concepts such as simple event, complex event, event equivalence class and event generator will be given in this section.

2.1 Simple Event

Events are real-world occurrences that unfold over space and time. In other words, an event is something notable that happens, owning a lasting time, occurs in a specific place, and typically will involve certain change of state. The formal

Table 1. Symbol and meaning of $Type_s$

Symbol	Meaning of $Type_s$
A	CPU utilization
B	Capacity of memory occupied
C_1	Number of clicking mouse's left buttons
C_2	Number of clicking mouse's right button
C_3	Amount of moving pixel of mouse
D	Number of pressing keyboards
EF	Network flow
F_p $(1 \leq p \leq 7)$	p^{th} front running process of monitored applications

definition about simple event and association rules to infer complex events are described in the following.

Definition 1. *Let E($type_s$,t_s,p_s,S_s) represent a simple event, where parameter $type_s$, t_s, and p_s represent type, time and place that a simple event occurs respectively. And S_s is an attribute set of different simple events, i.e.,S_s =(S_1,S_2,..., S_n), where S_i $(1 \leq i \leq n)$ is the i^{th} attribute in S_s.*

Since all simple computer-mediated interactive events occur in a common place, parameter p_s can be omitted in this paper. In computer-mediated interactions, a simple event's S_s has an unique attribute depending on $type_s$. Therefore, both $type_s$ and different values of S_s's unique attribute can act as a monitoring measure together. Because of this, simple event $E(type_s,t_s,p_s,S_s)$ can be simplified into the form $E(Type_s, t_s)$, where parameter $Type_s$ not only reflects an event type but also gives its measurable value. Further, in order to facilitate discussing, we will use abbreviation E to replace $E(Type_s, t_s)$ in the following unless otherwise specified.

We focus on 8 aspects of simple events relating to computer-mediated interactions, which are (1) CPU utilization, (2) capacity of memory occupied, (3) the number of clicking mouse's left buttons, (4) the number of clicking mouse's right buttons, (5) the amount of moving pixel of mouse, (6) the number of pressing keyboards, (7) network flow and (8) a front running process of monitored applications. In fact, since there are lots of applications in real computer world, it is unrealistic to monitor all applications. Therefore, for simplifying, only 7 typical processes are selected, denoted by F_p $(1 \leq p \leq 7)$, which are IE explorer, Google explorer, War3 (a real time strategy game), Trading Card Game Online (a board game), Windows Media Player, QQ (an instant message software) and MSN. The different values of $Type_s$ are listed in Table 1. Based on the 8 aspects mentioned above, there are totally 8 types of simple events need to monitor, which are listed in the following.

1. Once A is over 50 %, an instance of E, $e(A,t)$, is captured and created.
2. Once B is over 60 %, an instance of E, $e(B,t)$, is captured and created.
3. Once C_1 is over 30, an instance of E, $e(C_1,t)$, is captured and created.
4. Once C_2 is over 10, an instance of E, $e(C_2,t)$, is captured and created.

5. Once C_3 is over 1600, an instance of E, $e(C_3,t)$, is captured and created.
6. Once D is over 100, an instance of E, $e(D,t)$, is captured and created.
7. Once EF is over 40 MB, an instance of E, $e(EF,t)$, is captured and created.
8. Once F_p is running, one of instances of E, $e(F_p,t)$, is captured and created.

2.2 Complex Event

Let $cE(type_c, t_c, p_c, S_c)$ represent a complex event, where each parameter's subscript c distinguishes complex events from simple events. Above all, parameter p_c can be omitted due to the same reason as that of simple events. Second, similar with simple event's characteristic of unique attribute, it is enough for complex events to let S_c only record their lasting time. Thus, based on two points just mentioned, $cE(type_c, t_c, p_c, S_c)$ can be simplified into the form $cE(type_c, t_c, dur)$, where the first two parameters represent the type and time that a complex event occurs and the last parameter dur represents event's duration time. In addition, substitute $Type_c$ for $type_c$ in order to keep coincident with the type expression of simple events. As a result, $cE(type_c, t_c, dur)$ is written into $cE(Type_c, t_c, dur)$. The different values and meanings of parameter $Type_c$ are listed in Table 2.

Table 2. $Type_c$ and Weight

Time Interval	Meaning	Weight
$WVOn$	Watching Video Online	0.198
$WVOff$	Watching Video Offline	0.131
$PRTSG$	Playing Real Time Game	0.205
PBG	Playing Board Game	0.167
BWS	Browsing Web Site	0.155
COn	Chatting Online	0.143
DL	DownLoading	0.001

Table 3. T and Weight

Time interval	T	Weight
$6am-11am$	$morning$	0.14
$11am-14pm$	$noon$	0.16
$14pm-18pm$	$afternoon$	0.14
$18pm-23pm$	$evening$	0.23
$23pm-6am$	$before\ dawn$	0.33

The association rules for identifying complex events are represented in a disjunctive normal form. Let R denote association rules set, then $R = (r_1 \vee r_2 \vee ... \vee r_w)$, $(1 \leq w \leq 7)$, where $v_i's$ are the disjuncts. Each rule can be expressed in Formula (1).

$$r_v : (Condition_v) \rightarrow cE(Type_c^v, t, dur) \tag{1}$$

The left-hand side of the rule is called the rule antecedent or precondition. It contains a disjunctive normal of the conjunction of simple event tests, which is shown in Formula (2).

$$\begin{aligned} Condition_v = & [e(Type_s^1, t) \wedge e(Type_s^2, t) \wedge ... \wedge e(Type_s^m, t)] \\ & \vee [e(Type_s^1, t) \wedge e(Type_s^2, t) \wedge ... \wedge e(Type_s^h, t)] \\ & \vee ... \\ & \vee [e(Type_s^1, t) \wedge e(Type_s^2, t) \wedge ... \wedge e(Type_s^g, t)] \end{aligned} \tag{2}$$

Table 4. Association Rules of Generating Complex Events

r_1: $e(A, t) \wedge e(B, t) \wedge e(EF, t) \rightarrow ce(WVOn, t, dur)$
r_2: $e(A, t) \wedge e(B, t) \wedge e(EF_1, t) \rightarrow ce(WVOff, t, dur)$
r_3: $e(A, t) \wedge e(B, t) \wedge e(C_1, t) \wedge e(C_2, t) \wedge e(C_3, t) \wedge e(D, t) \wedge e(F_4, t)$
$\rightarrow ce(PRTSG, t, dur)$
r_4: $[e(A, t) \wedge e(B, t) \wedge e(C_1, t) \wedge e(C_3, t) \wedge e(F_3, t)]$
$\vee [e(A, t) \wedge e(C_1, t) \wedge e(C_3, t) \wedge e(F_3, t)] \rightarrow ce(PBG, t, dur)$
r_5: $[e(C_3, t) \wedge e(F_1, t)] \vee [e(C_3, t) \wedge e(F_2, t2] \vee [e(C_3, t) \wedge e(F_1, t) \wedge e(EF, t)]$
$\vee [e(C_3, t) \wedge e(F_2, t) \wedge e(EF, t)] \vee [e(C_1, t) \wedge e(C_3, t) \wedge e(F_1, t)]$
$\vee [e(C_1, t) \wedge e(C_3, t) \wedge e(F_2, t)] \vee [e(C_1, t) \wedge e(C_3, t) \wedge e(F_1, t) \wedge e(EF, t)]$
$\vee [e(C_1, t) \wedge e(C_3, t) \wedge e(F_2, t) \wedge e(EF, t)]$
$\rightarrow ce(BWS, t, dur)$
r_6: $[e(D, t) \wedge e(F_6, t)] \vee [e(D, t) \wedge e(F_7, t)] \vee [e(D, t) \wedge e(C_3, t) \wedge e(F_6, t)]$
$\vee [e(D, t) \wedge e(C_3, t) \wedge e(F_7, t)] \rightarrow ce(PBG, t, dur)$
r_7: $e(EF, t) \rightarrow ce(DL, t, dur)$

where $1 \leq (m,h,g) \leq 8$. Parameter m, g, h represent the number of simple events in a different conjunction normal. The right-hand side of the rule is called the rule consequent. If the precondition of r_v is satisfied, then r_v is said to be triggered, which results in the generation of $ce(Type_c^v, t, dur)$, i.e., an instance of a complex event. Association rules to deduce complex events are shown in Table 4. It is obvious that there are 7 association rules, which results in $1 \leq v \leq 7$. For example, if $e(A, t)$, $e(B, t)$ and $e(EF, t)$ are monitored simultaneously at time t, rule r_1 will be triggered. As a result, $ce(WVOn, t, dur)$ will be generated. In other words, we can induce that the computer user is watching video online at time t with dur=null because t is a time point. In order to obtain the duration time of $ce(Type_c^v, t, dur)$, an approach to obtaining complex event's lasting time will be exploited, which is introduced in next paragraph in detail.

During computing the lasting time of complex event, an interesting phenomenon is observed. Many complex events with a same event type are treated as different events just because their occurring time or lasting time is different. In fact, if time is limited to a reasonable range, these events have no obvious distinction. For example, given two complex events, "Surfing the Internet starting at 8:00 a.m. for 47 min" and "Surfing the Internet at 9:00 a.m. for 52 min", it is obvious that both of them have little semantic difference in real life, as they all happen in morning and the duration difference is not too much. However, two independent complex events are generated in event processing. This phenomenon results in the number of frequent complex events is too much, here, which is called Excessive Fragmentation of Time (EFT). For avoiding this issue, complex events can be merged together based on some coarse time granularity. Premise is this reduction has no negative effect on the precision of emerging pattern mining.

Considering the time semantic nature of real life, it is reasonable to partition a day with 24 h into 5 time intervals, i.e., (6:00 a.m. - 11:00 a.m.), (11:00 a.m. - 14:00 p.m.), (14:00 p.m. - 18:00 p.m.), (18:00 p.m. - 23:00 p.m.) and (23:00 p.m. -

6:00 a.m.) in this paper, and each of them represents "morning", "noon", "afternoon", "evening" and "before dawn" respectively. For avoiding EFT effects on both t_c and dur, two solution strategies, i.e., Occurrence Time Mapping Strategy (OTMS) and Duration Rounding Strategy (DRS), are proposed in this paper. The basic idea of OTMS is to merge some complex events into a group, where one to one mapping relationship between groups and 5 time intervals need to satisfy. For emphasizing the semantic meaning of occurrence time, parameter t_c is replaced with T_c. The basic idea of DRS is to let duration is the multiple times of integer 10, here, time unit is minute, and if not, round it. Both OTMS and DRS all decrease the number of complex events dramatically. Finally, complex event's abstract form $cE(Type_c, t_c, dur)$ is represented as $CE(Type_c, T, Dur)$. Table 3 lists 5 values of T in detail. In addition, in order to facilitate discussing, we will use abbreviation CE and Ce, to replace $CE(Type_c, T, Dur)$ and its instance respectively, in the following unless otherwise specified.

2.3 Equivalence Class and Generator of Complex Event

A CE dataset is a set of CE transactions. A CE transaction is a non-empty set of CEs. The key idea of PIU-Miner is to mine a concise representation of equivalence classes of frequent CE set from a transactional CE database D. Formally, an equivalence class of CEs is defined as follows.

Definition 2. *Let EC_{CE} represent an equivalence class of CEs. EC_{CE} is a set of CEs that always occur together in some CE transactions of D. That is, $\forall X, Y \in EC_{CE}, \exists f_D(X) = f_D(Y)$, where $f_D(Z) = \{R \in D \mid Z \subseteq R\}$.*

Definition 3. *The support of an CE itemset P_{CE} in a dataset D, denoted by sup(P_{CE}, D), is the percentage of CE transactions in D that contain P_{CE}.*

Definition 4. *Let X be a CE itemset of a CE dataset D. The equivalence class of X in D is denoted $[X]_D$. The maximal CE itemset and the minimal itemsets of $[X]_D$ are called the closed pattern and the generators of this CE equivalence class respectively. Generator of $[X]_D$ is represented as G. The closed patterns and generators of D are all the closed patterns and generators of their equivalence classes.*

Property 1. Let C_{CE} be the closed pattern of an equivalence class EC_{CE} and G_{CE} be a generator of EC_{CE}. Then all CE itemset X satisfying $G_{CE} \subseteq X \subseteq C_{CE}$ are also in this equivalence class.

Corollary 1. *An equivalence class EC_{CE} can be uniquely and concisely represented by a closed pattern C_{CE} and a set G_{CE} of generators, in the form of $EC_{CE} = [G_{CE}, C_{CE}]$, where $[G_{CE}, C_{CE}] = \{X \mid \exists g \in G_{CE}, g \subseteq X \subseteq C_{CE}\}$.*

Corollary 2. *The entire equivalence class can be concisely bounded as EC_{CE} = [G_{CE}, C_{CE}], where G_{CE} is the set of generators of EC_{CE}, C_{CE} is the closed pattern, and [G_{CE},C_{CE}] = $\{X \mid \exists g \in G_{CE}, g \subseteq X \subseteq C_{CE}\}$.*

In order to facilitating discuss, generator G_{CE} is abbreviated to G in the following unless otherwise specified.

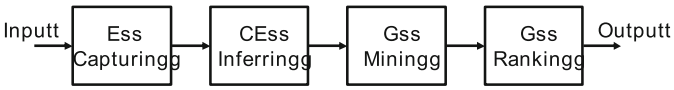

Fig. 1. PIU-Miner Model

3 PIU-Miner Model

PIU-Miner model, which is shown in Fig. 1, consists of four processing modules, 1) *Es* Monitoring Module for capturing $E(Type_s,t_s)$, 2) *CEs* Inferring Module for inducing $CE(Type_c,T,Dur)$ based on OTMS and DRS strategies, 3) *Gs* Mining Module for discovering PIU generators, and 4) *Gs* Ranking Module for scoring PIU generators to select optimal emerging patterns. In PIU-Miner model, user's interactions with computer are inputs and optimal PIU generators are outputs. Algorithm 1 further describes the total processing steps of PIU-Miner. Since *Gs* Mining Module is the core of PIU-Miner, Sect. 3.1 illustrated the detailed mining procedure. Section 3.2 gives the score formula to rank generators.

3.1 Discovering Generators of PIU

Two algorithms are introduced in this section, one is PIU Detecting Algorithm (PDA) and the other is Generator Mining Algorithm (GMA). The former illustrates the global processing steps and the latter focuses on how to mine generator, i.e., the lower bound of equivalence class of *CE itemset* in detail.

Algorithm 1. PIU Detecting Algorithm

Input: *SESet*, *CE*, *CESet*, *ESet*;
Output: *FG*

```
1: while (current time is out of sliding window) do
2:    while (simple event is not coming any longer) do
3:       ESet ← CapturingSimpleEvent();
4:       cE ← CreatingComplexEvent(ESet);
5:       insert cE to cESet;
6:    end while
7:    CESet ← MergingEvent(cESet);
8: end while
9: FG ← MiningGenerator(CESet);
10: OFG ← RankingGenrator(FG);
```

Given that *ESet* is a set storing simple event *E*, *cE* is a complex event inferred by *Es*, *cESet* is a set storing *cEs*, *CESet* is a set storing *CEs* processed by OTMS and DRS strategies, *FG* is the generator set, and *OFG* is the optimal generator set. PDA algorithm is given in Algorithm 1 and its processing

procedure includes 4 steps mainly. First, capture simple event set in last x time unit. Second, create a cE set based on simple event set captured. Third, cEs are mapped into CEs and their time durations are obtained. Finally, mine Gs based on the CEs resulting from merge processing.

GM algorithm is illustrated in Algorithm 2. Given that l is a frequent generator, D_l is the conditional database of l stored in an FP-tree [6] and FG is the set of Generators. Here, we use the modified FP-tree that the frequent events with a full support, i.e., those events appear every day, must be removed from the head table of FP-trees. After the modified FP-tree is constructed, the subsequent mining is operated on top of such tree. As an input of Mining Generator Algorithm, the details for frequent generator and conditional database are in [7].

Algorithm 2. Generator Mining Algorithm

Input: l, D_l
Output: FG
1: **for** l **do**
2: scan D_l to count frequent events and sort them into descending frequency order, denoted as $F_l = \{ a_1, a_2, ..., a_m \}$;
3: **for** $a_i \in F_l$ **do**
4: if sup($l \cup \{a_i\}$, D) = sup (l, D) then $F_l = F_l$ - $\{a_i\}$
5: **end for**
6: **for** $a_l \in F_l$ **do**
7: if $\exists\ l' \subset l \cup \{a_i\}$ and sup(l',D)= sup(l $\cup \{a_i\}$, D) then $F_l = F_l$ - $\{a_i\}$
8: else insert $l \cup \{a_i\}$ into FG
9: **end for**
10: **end for**

3.2 Ranking Generators of PIU

Since we have mined a set of generators FG, then our work is ranking these generators, identifying which one enables to represent PIU markedly. In this process, experts in PIU domain are involved in detecting work, scoring each one in generator set and rank the useful ones. According to knowledge and experiences, a formula for scoring every generator is given in Formula (3). Just like mentioned in Sect. 2.2, in Formula (3), $Type_c$ represents the type of frequent CE , T represents a CE's time interval projected by OTMS and Dur is CE's lasting time. The weights of $Type_c$ and T are listed in Table 2 and 3 respectively. Parameter k points out the number of CEs included in a generator G. After scoring all the generators, the generator set whose score less than threshold r_s will be pruned. The remaining ones are the final emerging patterns to represent PIU optimally.

$$Score(G_i\langle\bigcup_{k=1}^{n} CE_k\rangle) = \sum_{k=1}^{n} weight(Type_c^k) \times weight(T_k) \times Dur_k \qquad (3)$$

Table 5. The first week's Gs

G_1 ⟨ $Ce(COn, evening, 30)$, $Ce(BWS, before\ dawn, 20)$ ⟩
G_2 ⟨ $Ce(WVOn, evening, 80)$, $Ce(BWS, noon, 20)$ ⟩
G_3 ⟨ $Ce(WVOn, before\ dawn, 80)$, $Ce(COn, evening, 30)$ ⟩
G_4 ⟨ $Ce(WVOn, before\ dawn, 80)$ ⟩

4 Experiment Results and Evaluation

In this paper, PIU-Miner model is proposed for detecting PIU from computer-mediated interaction events. For testing the performance of PIU-Miner model, extensive experiments are exploited in this section. All the experiments are done with Intel Core i3 processor (2.53 GHz CPU with 4GB RAM) and operating system is Windows 7 professional edition.

Among 20 students working in our college, we select 5 persons who have high PIU probability such as watching video online, playing board game, and Browsing Web Site by questionnaires as testing objects. We collected their daily computer interactions about 10 weeks as sample data set. For easy to understand generators, an example of generators mined by PIU-Miner model, i.e., the first week's Gs, is shown in Table 5. PIU-Miner model discovers 4 generators for detecting PIU in the first week, which tell us some clues about a person who has some PIU possibilities of watching video online, chatting on line and browsing web site frequently.

Since threshold r_s is a key factor that effects the precision of generators, precision values as threshold varies are tested and is shown in Fig. 2. Every precision measure is the average values of 10 weeks' accuracy data. It is obvious that r_s=3.5 is the best selection for our testing sample data. Threshold r_s=3.5 means that the generators will be pruned if their scores are less than 3.5. Based on threshold 3.5, we statistic the precision measures for detecting PIU based on discovered generators every week and then give the corresponding results in Fig. 3. This histogram graph indicates the best precision is up to 0.84 and the lowest precision is 0.6. Figure 3 proves that the generators mined by PIU-Miner are effective.

5 Related Work

Discovering frequent patterns from large database is meaningful and practical in association rule mining [8,9], correlation analysis [10], classification [11], emerging pattern [12] and other domains. Some frequent pattern approaches, like Apriori [13] and FP-growth [7], base on a single minimum support (minsup), thus confront a dilemma when mining frequent patterns consisting of both frequent and rare items. That is, at high minsup, frequent patterns involving rare items will be missed, and at low minsup, the number of frequent patterns explodes. To solve this problem, the approach using multiple minsup constraints is raised in [14–18]. There are mainly three types of mining frequent patterns approach.

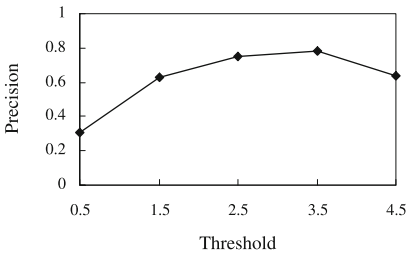

Fig. 2. Precision of Gs under r_s

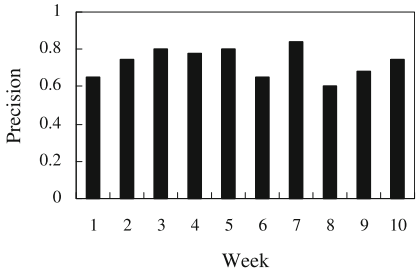

Fig. 3. Precision of 10 weeks' Gs

The first is traversing iteratively the set of all patterns. The most prominent algorithm based on this approach is Apriori and many improved algorithms [19–22] arose later. The second is extracting maximal frequent patterns of which all supersets are infrequent and all subsets are frequent. The most prominent algorithm based on this approach is Max-Miner [23]. The third is based on the theoretical framework [24] that uses the closure of the Galois connection [25]. The most representative is Close algorithm [26].

Sequential pattern mining was first introduced by Agrawal and Strikant [27] used in data mining research field. In sequential pattern mining process, the algorithm only mines the closed patterns instead of all frequent ones, for the former leads to a complete yet compact result thus better efficiency. Agrawal *et al.* [28] developed a generalized and refined algorithm, GSP, based on the Apriori property [8]. Since then, many improved algorithm have been proposed. SPADE [29], PrefixSpan [30] and SPAM [31] are quite popular ones. All these search strategies can be divided into two types [32] - breadth-first search, such as GSP and SPADE, generating many candidate patterns, and depth-first search, such as PrefixSpan and SPAM, iteratively partitioning the original data set. While most of previously developed closed pattern mining algorithms are inherently costly in both runtime and space usage when the support threshold is low or the patterns become long, Wang *et al.* [33] presented an efficient algorithm for mining frequent closed sequences, BIDE, without candidate maintenance. In recent years, the studying domain of sequential pattern mining has been extended. Since existing work of studying the problem of frequent sequence generator mining is rare, Gao *et al.* [34] present a novel algorithm, FEAT (abbr. frequent sequence generator miner), to perform this work. Yan *et al.* [35] studied the closed pattern mining problem and proposed the CloSpan algorithm. Zhang *et al.* [36] studied the sequential pattern mining from a single long sequence with gap constraints. Sequential pattern mining is widely used in many domains. For instance, in text mining, sequential patterns are mined in a single document and a document collection. It is also successfully applied to question answering [37] and authorship attribution extraction [38].

Internet addiction was first introduced by Young [39], but the illustrate of term addiction is various between scholars and has greatly developed. In paper [1], Young defined Internet addiction as impulse-control disorder by using Pathological Gambling as a model. The behavior was first named as Pathological Compulsive Internet Usage (PCIU) and later changed to Pathological Internet Use (PIU), divided into 5 types - information overload, net compulsions, cyber-relationship addiction, cyber-sexual addiction and game addiction [40]. There are two approaches for PIU diagnosis. One is based on The Diagnostic and Statistical Manual of Mental Disorders (DSM). A representative is Coldberg's idea. He came up with 6 criteria according to DSM and anyone who reaches these criteria can be diagnosed as PIU [1]. Another is based on cognitive-behavioral criteria. Davis believes that cognitive-behavior should be considered in PIU diagnosis and he divides PIU into two types - Specific Pathological Internet Use (SPIU) and Generalized Pathological Internet Use (GPIU), thus can treat characteristic and generality in a different way [41].

6 Conclusion and Future Work

Be absorbed in Internet so addictively as to unable to control, this phenomenon is called PIU or IAD. PIU is a negative production of Internet popularizing and need to avoid. Aiming at this issue, a novel PIU-Miner model is proposed in this paper. The basic idea of PIU-Miner is to mine Generators only (i.e., a minimum event pattern set), other than all possible event emerging patterns, for detecting PIU group from Internet users. Extensive experimental results show that Generators-based event pattern discovering method for diagnosing PIU is efficient and effective.

Future work will focus on how to build PIU classifiers based on mined generators. For achieving this motivation, we will take domain knowledge such as medical science, psychology and sociology into account, which would be helpful to improve the accuracy of classifiers further.

References

1. Goldberg: Internet addiction disorder. http://www.psycom.net
2. Young, K.: Internet addiction: symptoms, evaluation and treatment. J. Innov. Clin. Pract. **17**, 19–31 (1999)
3. Young, K.: Caught in the Net. Wiley, New York (1998)
4. Beard, W., Wolf, M.: Modification in the proposed diagnostic criteria for internet addiction. J. Cyberpsychol. Behav. **3**, 377–383 (2001)
5. Cao, L., Gorodetsky, V., Mitkas, P.: Agent mining: the synergy of agents and data mining. IEEE Intell. Syst. **24**(3), 64–72 (2009)
6. Li, J., Liu, G., Wong, L.: Mining statistically important equivalence classes and delta discriminative emerging patterns. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 430–439 (2007)

7. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. J. Data Min. Knowl. Disc. **8**(1), 53–87 (2004)
8. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, pp. 487–499 (1994)
9. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Proceedings of the Third International Conference on Information and Knowledge Management, pp. 401–408 (1994)
10. Brin, S., Motwani, R., Silverstein, C.: Beyond market basket: generalizing association rules to correlations. In: Proceedings ACM SIGMOD International Conference on Management of Data, pp. 265–276 (1997)
11. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 80–86 (1998)
12. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 43–52 (1999)
13. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
14. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 337–341 (1999)
15. Lee, Y., Hong, T., Lin, W.: Mining association rules with multiple minimum supports using maximum constraints. Int. J. Approx. Reason **40**(1–2), 44–54 (2005)
16. Hu, Y., Chen, Y.: Mining association rules with multiple minimum supports: a new algorithm and a support tuning mechanism. J. Decis. Support Syst. **42**(1), 1–24 (2006)
17. Uday Kiran, R., Krishna Reddy, P.: An improved multiple minimum support based approach to mine rare association rules. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, pp. 340–347 (2009)
18. Uday Kiran, R., Krishna Reddy, P.: An improved frequent pattern-growth approach to discover rare association rules. In: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, pp. 43–52 (2009)
19. Brin S., Motwani R., Ullman J., Tsur S.: Dynamic itemset counting and implication rules for market basket data. In: Proceedings ACM SIGMOD International Conference on Management of Data, pp. 255–264 (1997)
20. Park, J., Chen, M., Yu, P.: An efficient hash based algorithm for mining association rules. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 175–186 (1995)
21. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. In: Proceedings of 21th International Conference on Very Large Data Bases, pp. 432–444 (1995)
22. Voivonen, H.: Sampling large databases for association rules. In: Proceedings of 22th International Conference on Very Large Data Bases, pp. 134–145 (1996)
23. Bayardo, R.: Efficiently mining long patterns from databases. In: Proceedings ACM SIGMOD International Conference on Management of Data, pp. 85–93 (1998)
24. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Pruning closed itemset lattices for association rules. In: BDA, pp. 177–196 (1998)

25. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (1999)
26. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. J. Inf. Syst. **24**(1), 25–46 (1999)
27. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14 (1995)
28. Srikant, R., Afrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th International Conference on Extending Database Technology, pp. 3–17 (1996)
29. Zaki, M.: Spade: an efficient algorithm for mining frequent sequences. J. Mach. Learn. **42**, 31–60 (2001)
30. Pei, J., Han, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: mining sequential patterns by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224 (2001)
31. Ayresm, J., Gehrke, J., Yiu, T., Flannick, J.: Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 429–435 (2002)
32. Feng, J., Xie, F., Hu, X., Li, P., Cao, J., Wu, X.: Keyword extraction based on sequential pattern mining. In: The Third International Conference on Internet Multimedia Computing and Service, pp. 34–38 (2011)
33. Wang, J., Han, J.: BIDE: efficient mining of frequent closed sequences. In: Proceedings of the 20th International Conference on Data Engineering, pp. 79–90 (2004)
34. Gao, C., Wang, J., He, Y., Zhou, L.: Efficient mining of frequent sequence generators. In: Proceedings of the 17th International Conference on World Wide Web, pp. 1051–1052 (2008)
35. Yan, X., Han, J., Afshar, R.: CloSpan: mining closed sequential patterns in large datasets. In: Proceedings of the Third SIAM International Conference on Data Mining, pp. 166–177 (2003)
36. Zhang, M., Kao, B., Cheung, D., Yip, K.: Mining periodic patterns with gap requirement from sequences. J. ACM Trans. Knowl. Discov. Data **1**(2) (2007)
37. Denicia-Carral, C., Montes-y-Gómez, M., Villaseñor-Pineda, L., Hernández, R.G.: A text mining approach for definition question answering. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 76–86. Springer, Heidelberg (2006)
38. Coyotl-Morales, R.M., Villaseñor-Pineda, L., Montes-y-Gómez, M., Rosso, P.: Authorship attribution using word sequences. In: Martínez-Trinidad, J.F., Carrasco Ochoa, J.A., Kittler, J. (eds.) CIARP 2006. LNCS, vol. 4225, pp. 844–853. Springer, Heidelberg (2006)
39. Young, K.: Internet addiction: the emergence of a new clinical disorder. J. Cyberpsychol. Behav. **1**(3), 237–244 (1996)
40. Young, K.: Internet addiction: a new clinical phenomenon and its consequences. J. Am. Behav. Sci. **48**, 402–415 (2004)
41. Davis, R.: A cognitive-behavioral model of pathological internet use. J. Comput. Hum. Behav. **17**(2), 187–195 (2001)