Monitoring Partial Updates in Web Pages using Relational Leaning

Seiji YAMADA and Yuki NAKAI

CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama, 226-8502, Japan
yamada@ymd.dis.titech.ac.jp

Abstract. This paper describes an automatic monitoring system that constantly checks partial updates in Web pages and notifies them to a user. While one of the most important advantages of the WWW is frequent updates of Web pages, we need to constantly check them out and this task may take much cognitive load. Unfortunately applications to automatically check such updates can not deal with partial updates like updates in a particular cell of a table in a Web page. Hence we developed a automatic monitoring system that checks such partial updates. A user can give a system regions in which he/she wants to know the updates in a Web page as training examples, and it is able to learn rules to identify the partial updates by relational learning. We implemented the system and some executed examples were presented.

1 Introduction

We currently obtain various information from the WWW and utilize them. While one of the most important advantages of the WWW is its constant updates of Web pages, we needs to frequently check the updates for acquiring the latest information and this task forces much cognitive load on us. Thus a number of applications to automatically check and notify updates of Web pages have been developed[1][2]. Unfortunately almost all of them notify updates to a user whenever any part of a Web page is updated, and most of such updates may not useful to him/her.

Consider a weather report Web page and a user who has a plan to go to a picnic on the next Sunday and is interested in the weather. He/she needs to frequently check the next Sunday's weather in the Web page. If a user employs a Web update checking application, it notifies him/her all of updates including other day's weather changes except Sunday thought such notifications are meaningless. Thus *partial update* is defined as an update of a region in which a user is interested, not of any part of a Web page. We consider this partial update monitoring is widely necessary in a lot of fields like stock market pages, the exchange rate pages and so on.

We developed an automatic monitoring system PUM (Partial Update Monitoring) that constantly checks partial updates in Web pages and notifies them to a user.

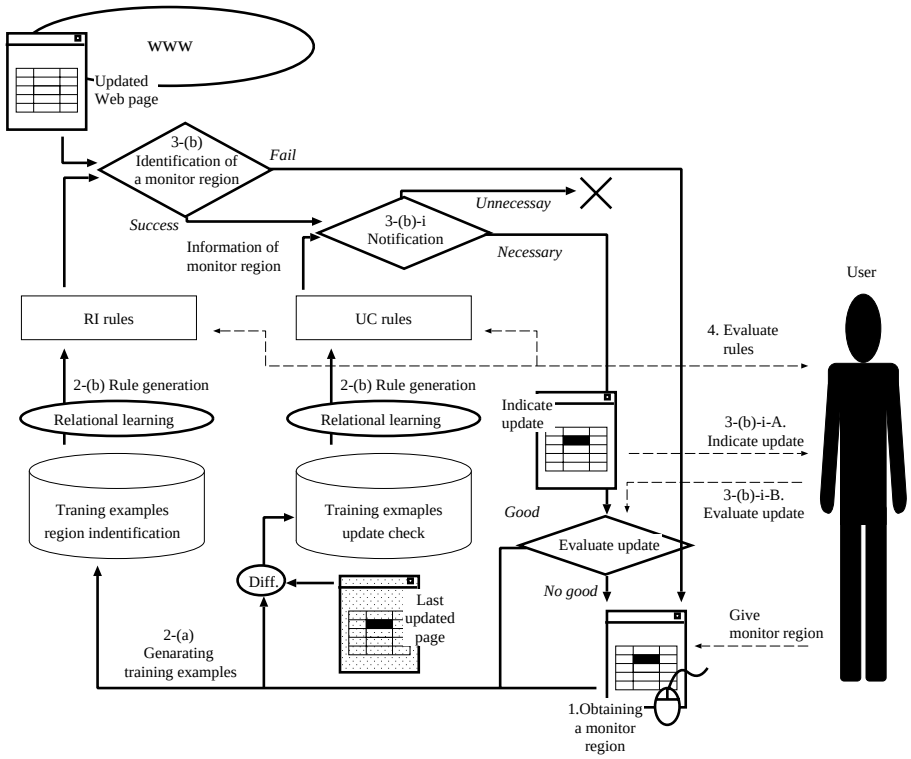

Fig. 1. System overview.

2 PUM: partial update monitoring in a Web page

2.1 System overview

PUM is a system that identifies a region indicated by a user in a Web page, checks partial updates in the region and notifies a user the updates which he/she wants to know. Fig.1 shows overview of PUM , where a dotted line indicates interaction between a user and PUM .

Fig.1 also stands for the procedure of PUM . First of all, PUM obtains a monitor region from a user. A user indicates a region in which he/she wants to know the update by mouse highlight operation on interface of PUM (Fig.2). Next PUM extracts training examples for both of RI(region identification) and UC(update check) from a region indicated by a user.

Then a relational learning system automatically acquires two kinds of rules for region identification and update check. PUM utilizes RIPPER[3] as a relational learning system. RIPPER acquires rules to classify examples into two classes, and the learned rule is described as with symbolic representation.

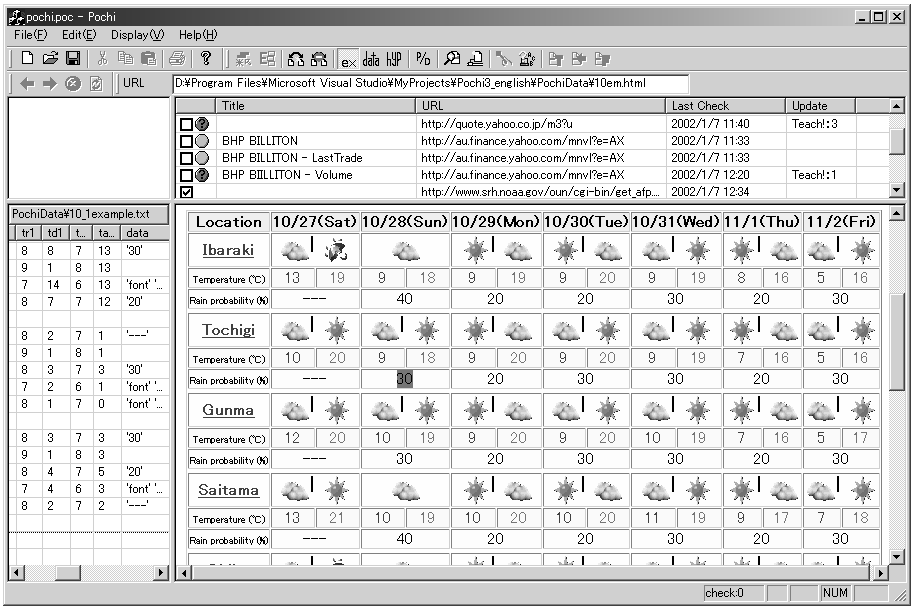

Fig. 2. Interface of PUM .

After such rules were generated, PUM becomes able to identify partial an update and determine whether it is one which a user wants to know or not by using two kinds of rules. If PUM decides an update is useful to a user, it notifies the update to a user. Otherwise PUM indicates the updated Web page to a user and obtains his/her evaluation. PUM is implemented using Visual C++ and Ruby on Windows2000.

Fig.2 shows interface of PUM . The window consists of three sub-windows: a Web browser window, an URL window and a training example window. A Web browser window (lower right in Fig.2) shows a Web page in the same way to Web browser and a user can easily indicate a region by highlighting it using a mouse. An URL window(upper in Fig.2) stand for URLs of updated pages. A training example window(lower left in Fig.2) indicates a table of attribute and value of stored training examples.

2.2 Negative examples for region identification

Since relational learning is a kind of inductive learning, negative examples play a important role to avoid over-generalization. Thus PUM automatically generates negative example for region identification to improve learning efficiency.

We consider neighborhood of an indicated region are near miss examples. Hence PUM generates negative examples from four regions: left, right, upper and lower regions to an indicated region.

Table 1. RI rules.

Eval. No.	Class	Condition
1	Good	cIndex ~'10/14(Sun)', rNo ~'7'.
2	Good	rNo ~'7', cIndex ~'Sun'.

3 Executed examples

A typically successful example for PUM is on updates in a weather report Web page shown in Fig.2. This page shows weather report of next seven days in which a table is scrolled horizontally. In this example, a user wants PUM to notify an update when rain probability of Tochigi on Sunday (a highlighted cell in Fig.2) decreases less than 40%. Thus PUM needs to learn RI rules to identify a cell indicating weather probability of Tochigi on Sunday and UC rules to check the value of weather probability is less than 40. PUM successfully became able to extract the correct partial update after several evaluations by a user.

Table1 stands for the number of user's evaluations and learned RI rules at that time. A rule consists of "Class" and "Condition", and if an update satisfies "Condition", it is classified into the "Class". $A\tilde{}B$ in "Condition" means a condition that B is included in a attribute A. A RI rule learned from the first evaluation can identify a cell which is in 7th-row and has '10/14(Sun)' as a column index. This rule succeeded in identifying a region for four days, however it failed on fifth day. Because a target region included '10/21(Sun)' instead of '10/14(Sun)' by scrolling. Then PUM requires second user's evaluation and learned a new rule shown in Table1. This second rule identifies a correct cell using more general condition 'Sun' as a column index, not '10/14(Sun)'.

Additional successful examples were investigated in stock market, CD ranking, exchange rate Web pages and so on.

4 Conclusion

We proposed a monitoring system PUM that constantly checks partial updates in Web pages and notifies them to a user. A user can give a system regions which he/she wants to know the updates in a Web page as training examples, and it can learn rules to detect the partial updates by relational learning. We implemented our system and some executed examples were presented.

References

1. Web Secretary. (http://homemade.hypermart.net/websec/)
2. Saeyor, S., Ishizuka, M.: WebBeholder: A revolution in tracking and viewing changes on the web by agent community. In: WebNet 1998. (1998)
3. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning. (1995) 115–123