Towards Autonomic Web-sites Based on Learning Automata

Pradeep S
Indian Institute of Information Technology
26/C, Electronics City, Hosur Road
Bangalore, INDIA 560100
pradeep.s@iiitb.ac.in

Chitra Ramachandran
Indian Institute of Information Technology
26/C, Electronics City, Hosur Road
Bangalore, INDIA 560100
chitra.ramachandran@iiitb.ac.in

Srinath Srinivasa
Indian Institute of Information Technology
26/C, Electronics City, Hosur Road
Bangalore, INDIA 560100
sri@iiitb.ac.in

ABSTRACT

Autonomics or self-reorganization becomes pertinent for web-sites serving a large number of users with highly varying workloads. An important component of self-adaptation is to model the behaviour of users and adapt accordingly. This paper proposes a learning-automata based technique for model discovery. User access patterns are used to construct an FSM model of user behaviour that in turn is used for prediction and prefetching. The proposed technique uses a generalization algorithm to classify behaviour patterns into a small number of generalized classes. It has been tested on both synthetic and live data-sets and has shown a prediction hit-rate of up to 89% on a real web-site.

Categories and Subject Descriptors: I.2.6 Computing methodologies Artificial Intelligence [Learning], I.5.1 Computing methodologies Pattern Recognition [Models]

Keywords: Autonomic website, Learning automata, Generalization

1. INTRODUCTION

Large websites are faced with the problem of highly varying user loads. A similar problem occurs in a proxy server, where documents need to be prefetched to improve performance. Both these issues involve the problem of "user modeling." Most of the currently existing techniques have the following issues: they either require supervision for learning user behaviour and/or ignore the history with which a given page is reached. In our approach, a technique based on learning automata (also known as *Grammatical Inference (GI)*) is proposed. The learning-automata is modeled as a Finite State Machine (FSM) based on a set of observed strings which can explain the *general class* of behaviours exemplified by the strings.

Grammatical inference is a technique of inferring the grammar of a language given a few sample strings. An important aspect of GI is *generalization*. A model is much smaller (and hence more "general") than the unfolded set of behaviours that it can generate. The generalization method proposed is an extension of a method called "Shortest Run Generalization Algorithm (SRA)" proposed in [3].

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

2. EXTENDED SRA

In the extended SRA, generalization is based on the following rule: If a pattern r repeats in the range $[p, p+k]$, where p is the observed minimum number of occurrences for the pattern and k is a configurable parameter; then generalize it to the regular expression: $(r^p \cdot r^*)$.

When the first string is encountered, an FSM is constructed by embedding hypothetical states on either sides of all symbols. The transition symbols between the states are the symbols encountered in the string. For each edge t, its transition probability (denoted by p_t) is set to 1. For each edge t, the count of the number of strings traversing t (denoted by c_t) is also set to 1.

For subsequent strings the FSM construction process is as follows:

1. Begin from the start state and traverse the state machine based on the input sequence as long as there is a path from the current state on the given alphabet. If there is no corresponding transition, branch out of the FSM and create new states and corresponding transitions. For each transition t increment c_t by 1
2. For each transition $t = (s_a, s_b), s_a, s_b \in S$ calculate transition probability as $p_t = \frac{c_t}{\Sigma c_{(s_a,*)}}$, where $(s_a, *)$ denotes the set of all edges from state s_a to any other state.

Once the new string is assimilated into the FSM by the above steps, the FSM is either generalized immediately (called "generalize as you go") or the generalization is performed after all input strings have been assimilated (called "generalize at the end"). In either case, the generalization algorithm that is used is the same as explained below.

Extended SRA: The process of generalization comprises of two algorithms: a "backward sweep" followed by a "forward sweep". The backward sweep is as follows:
Algorithm: Backward sweep:

1. Combine the different accepting states generated by the different strings into a single state f.
2. A p-tail represents a suffix of any path of length p leading to the end state. If there are k or more p-tails leading into f, merge all corresponding states of these tails. Here, k is a configurable parameter called the *generalization threshold* that determines when a set of observations is "good enough" to generalize.

3. Suppose $s \xrightarrow{a} s_1$ and $s' \xrightarrow{a} s'_1$ are two transitions in the FSM where s is to be merged with s' and s_1 is to be merged with s'_1. In such cases, set the count of the new transition to be the sum of the count of each individual merged transitions. Hence $c_{(\{s,s'\},\{s_1,s'_1\})} = c_{(s,s_1)} + c_{(s',s'_1)}$.

4. Proceed backwards on each incoming edge and jump to step 2 to look for any more tails to merge.

When states are merged backwards, it may introduce non-deterministic transitions, where a single symbol may have more than one outgoing edge from a given state. The non-determinism in the transitions are eliminated in the forward sweep.

Algorithm: Forward sweep:

1. Start from the start state and traverse the FSM in a depth first fashion

2. For each state s having a non-deterministic transition on a given input symbol a, merge all the states reachable by reading a in s.

3. For every merged transition $t = t_1 + t_2$, set the count $c_t = c_{t_1} + c_{t_2}$

After the forward sweep is performed, probabilities of each transition are calculated based on the new count values. The forward sweep may sometimes introduce states having tails crossing the generalization threshold k. These are not removed immediately, but are handled whenever the FSM is generalized the next time.

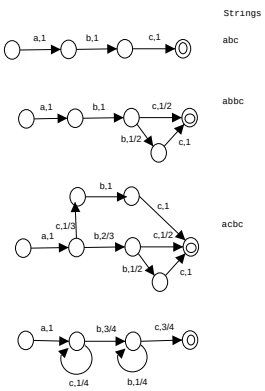

Figure 1: Generalization with $k = 3$

Figure 1 shows a generalization process with a threshold k set to 3. Generalization does not happen until at least 3 evidences are found for a given pattern. A more detailed treatment of the SRA algorithm alongwith algorithm analysis, correctness proofs and literature survey may be found in [2].

3. PREFETCHING AND CACHING BASED ON LEARNING AUTOMATA

An important performance related issue in proxy servers is to predict user behaviours and perform prefetching and caching of web pages that the users are likely to visit. A prefetching scheme based on the SRA algorithm has been implemented in a proxy server.

The algorithm was tested on both synthetic and real-life data. Performance analysis on live data was carried out on two datasets: (a). On benchmark test logs provided by Perkowitz et al [1], (b). On traversal logs from the proxy server under the domain www.iiitb.ac.in.

The first data set was used as is, while the second data set was cleaned to remove requests from robots, viruses, etc. A detailed description of the performance experiment is available from [2]. Effectiveness of the state machine was measured by predicting the next step of user behaviour at each step and calculating the "hit ratio" of our predictions. The hit ratio of a user session is the ratio of number of correct predictions and the total number of links the user traversed in the session.

Figure 2 shows how the hit ratio changed as a function of the number of user sessions processed for the two different scenarios.

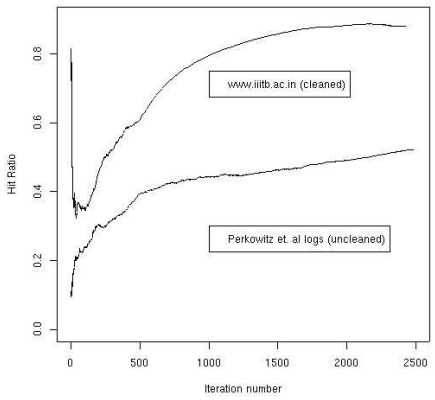

Figure 2: Average Hit ratio Vs Number of Strings

4. CONCLUSIONS

SRA presents a simple algorithm that can build usage models on the fly, in polynomial time. Experiments on real-life data sets have yielded promising results. However, adequate configuration needs to be performed to the learning machine before deployment in a real-world context. As shown by the performance results, accuracy of the generated model over a cleaned data set can be much higher than over an uncleaned data set.

5. REFERENCES

[1] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically learning from user access patterns. In *http://www.cs.washington.edu/research/adaptive/*.

[2] S. Pradeep, C. Ramachandran, and S. Srinivasa. Extended shortest-run generalization for web-site autonomics. *Technical Report, Open Systems Lab, IIIT Bangalore*, OSL-IIITB-0502, 2005.

[3] S. Srinivasa and M. Spiliopoulou. Discerning behavioural properties by analysing transaction logs. In *Proc of ACM Symposium on Applied Computing (SAC 2000), Como, Italy*, 2000.