An Extension of Association Rule Mining for Software Engineering Data Repositories

Shuji Morisaki, Akito Monden, Haruaki Tamada,
Tomoko Matsumura, and Ken-ichi Matsumoto
Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, Japan

Abstract

This paper proposes a method to mine rules from software engineering data repositories that contain a number of quantitative attributes such as staff months and SLOC. The proposed method extends conventional association analysis methods to treat quantitative variables in two ways: (1) the distribution of a given quantitative variable is described in the consequent part of a rule by its mean value and standard deviation so that conditions producing the distinctive distributions can be discovered. To discover optimized conditions, (2) quantitative values appearing in the antecedent part of a rule are divided into contiguous fine-grained partitions in preprocessing, then rules are merged after mining so that adjacent partitions are combined.

1. Introduction

Many software development companies collect data from software projects (records of product size, development duration, staff-hours, numbers of bugs, metrics for risk assessment, customer satisfaction, and the like), with the goal of improving productivity, meeting deadlines, and improving quality in software development. Generally, companies collect and store such software engineering data for use by production engineering divisions, quality assurance divisions, project management offices (PMOs), and other support divisions. Companies may use this information for purposes such as estimating developer effort, predicting reliability, and determining a wide range of development standards (such as bug density and productivity). For such purposes, a number of conventional analysis methods have been widely researched, including cost models [3] [11] [14], reliability models [9], and orthogonal defect classification [4].

This paper focuses on a new approach to association analysis utilizing the software project data described above. Researchers have used association analysis [1] effectively in the past to analyze point-of-sales (POS) data for retailers and Website traffic logs, to discover association rules hidden amongst the data [15]. There has also been research on software project data: through association analysis, Amasaki et al [2] mined preconditions for software projects to fall into disorder (combinations of risk assessment values) using the assessments of large numbers of risk variables.

General association analysis methods and rules, however, are not always applicable to software project data because they do not provide for scalar values. The values in software project data generally mix nominal measurements along with ordinal and scalar measurements, and it is therefore not possible to handle these values in a uniform fashion as-is. Software project data contains a number of quantitative measurements of particular interest so we would like to extend the general association analysis approach to take advantage of the scalar values instead of simply reducing them to nominal values. Identifying relationships among these values can lead to improved productivity, reduced bug density, and process improvements, as well as elimination of defect causes. Using their means and variance can help to more finely tune process improvements and cause identification. Finding a rule that identifies situations associated with higher bug density may make it possible to eliminate the causes of these bugs by eliminating the situations expressed by the rule. Similarly, finding rules common to projects with great amounts of variance in productivity may make it possible to reduce the variance by eliminating the situations common to the rules.

This paper proposes a method for mining rules appropriate for software project data by extending conventional association analysis methods. To handle staff-months, LOC, and other quantitative

variables, the proposed method extends association rules to include quantitative variables in the consequent parts of the rules. The proposed method divides these variables into contiguous fine-grained partitions for the antecedent parts of the rules. After mining extended association rules, the method merges rules by joining partitions next to each other.

In Section 2, below, we describe conventional association analysis and the issues for applying conventional association analysis to software project data. In Section 3 we describe the proposed method. Section 4 presents related research. Section 5 summarizes the findings and describes future topics.

2. Association Analysis and Its Issues

2.1 Association Analysis

Researchers have used association analysis to discover associations hidden amongst data in the POS product-purchasing logs of retail stores [1], Website traffic logs [15], proteins [10], and the like. For example, in the case of POS logs, researchers have mined rules about products purchased together, such as "purchases product A ∧ purchases product B ⇒ purchases product C." There are a number of possible uses for the rule in this example: the retailer could place products A, B, and C near to each other in the store so that customers can find them easily; or, it could ensure revenues by setting the prices of antecedent products A and B to make up the discounts on the sale price of consequent product C.

Association analysis is defined as follows [1].

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of binary attribute values, called items. A set $A \subset I$ is called an item set. Let a database D be a multi-set of I. Each $T \in D$ is called a transaction. An association rule is denoted by an expression $A \Rightarrow B$, where $B = I_k (1 \le k \le m)$, $A \cap B = \phi$

With data like POS logs, however, which have huge numbers of items, it is not realistic to mine all rules: it takes inordinate amounts of computer processing time, and it is not feasible to interpret the huge number of mined rules manually. For this reason, conditions are placed on rule mining, setting minimum values for one or all of three key indicators of rule importance (support, confidence, and lift). Rules that are not likely to be important are generally pruned.

Support:

Support is an indicator of rule frequency. It is expressed as $support(A \Rightarrow B)$, and is $support(A \Rightarrow B) = a/n$, where $a = \left|\{T \in D \mid A \subset T \wedge B \subset T\}\right|$ and $n = \left|\{T \in D\}\right|$.

Confidence:

Confidence is the probability that consequent B will follow antecedent A. It is expressed as $confidence(A \Rightarrow B)$, and is $confidence(A \Rightarrow B) = a/b$, where a is defined as in Support and $b = \left|\{T \in D \mid A \subset T\}\right|$.

Lift:

Lift is an indicator of the contribution antecedent A makes to consequent C. It is expressed as $lift(A \Rightarrow B)$, and is $lift(A \Rightarrow B) = confidence(A \Rightarrow B)/c$, where $c = \left|\{T \in D \mid B \subset T\}\right|$.

For example, assume that the number of projects, n = 20, the number of projects that contains A is 10, the number of projects that contains B is 8, and the number of projects that contains both A and B is 6. For A=>B, the support is 0.3 (6/20), the confidence is 0.6 (6/10), and the lift is 1.5 (0.6/8/20).

2.2 Issues with Association Analysis for a Software Engineering Data Repository

This paper envisions collecting software engineering data as the project progresses, and assumes that attributes include values such as staff effort and LOC as defined in the ISBSG repository [8] and IPA SEC [7]. Table 1 shows sample project data. In Table 1, row 1 is the attribute category, and row 2 is the attribute name. Each of the rows 3 and beyond corresponds to a single project. Many attribute values are measured and logged for each project. Note that all values in the table are made-up examples. Although the number of variables per project will differ depending on the organization and projects in question, there will be several hundred or so. On the other hand, there will be roughly from several tens to several thousands of projects. A company rarely has more than 10,000 projects. As shown in Table 1, a major characteristic of software project data is the existence of such nominal measurements as platform type, target industry, and target process, such ordinal measurements as performance requirements and security, and such scalar measurements as source lines of code (SLOC) and staff-hours (human costs).

Association analysis normally is applied to qualitative variables (nominal or ordinal measurements); scalar measurements are generally converted to ordinal measurements via preprocessing. For example, it would be possible to convert SLOC into an ordinal measurement consisting of three categories – high, medium, and low – depending on its value, but the optimum partition must be determined via trial and error, and it is a nontrivial task to discover the optimum partition points for multiple variables.

Table 1. An example of software development project data

Management Attributes		Project Attributes			Architecture			Requirements			Size				
Project ID	Dept. code	Development Type	Business Area Type	Application Type	Platform	Job	Database	Capability	Security	Portability	SLOC (Planned)	SLOC (Recorded)	Effort (Planned)	Effort (Recorded)	…
06S101	Industrial Dept. 1	New Development	Finance	Customer management	Windows	Interaction	DB2	Medium	High	N/A	10000	14239	12 staff month	16 staff month	…
06S201	Industrial Dept. 2	Re Development	Retail	Ordering	UNIX	Batch	Oracle	High	High	Low	28000	30940	60 staff month	68 staff month	…
06G01	Public Work Dept.	Enhancement	Government	Personnel affairs	Windows	Interaction	My SQL	Medium	High	Medium	8000	7900	12 staff month	8 staff month	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Sometimes, the variables in the software project data that most interest us in our analysis are quantitative variables. The variables that interest us are the ones that tie in directly to process improvement and elimination of defect causes. Some examples are productivity (ratio of LOC or FP to staff-hours worked), bug density, bugs detected per test case, and rate of outsourcing of the coding and testing phases. If we can discover conditions (rules) for changing values or distributions that have undesired impact, we can create countermeasures to the conditions. Below, we describe how the proposed method handles quantitative variables (scalar measurements) contained in the target data.

3. Extension of Association Rule

3.1 Preliminary Definitions

Each value in Table 1 is expressed as an <*attribute, value*> pair. Let projects be a set $P = \{P_1, P_2, ..., P_n\}$, and $P_i = \{< attr_1, p_{i1} >, < attr_2, p_{i2} >, ..., < attr_m, p_{im} >\} (1 \le i \le n)$, where $attr_k$ is the k^{th} attribute. P_i corresponds to the value of the k^{th} attribute. Further, let values of an attribute be a set $V = \{V_1, V_2, ..., V_m\}$, and $V_k = \{< attr_k, v_{1k} >, < attr_k, v_{2k} >, ..., < attr_k v_{n_k k} >\}$. Here, $v_{ik} \in V_k (1 \le i \le n_k)$ are either qualitative variables (nominal or ordinal measurements) or quantitative variables (scalar measurement). Note that in the case of quantitative variables/ordinal measurements, $v_{ik} < v_{ik+1}$.

Using Table 1 as an example, the third row in the table (the item with project ID 06S101) is P_1, and P_1 = {<project ID, 06S101>, <dept. code, industrial dept.1>, <development type, new development>, <business area type, finance>,…}. $attr_1$ is *project ID*, p_{11} is "06S101," and V_{14} = {<effort (planned), 12>, <effort (planned), 60>,…}, and $v_{1\,14}$ is "12."

3.2 Handling Quantitative Variables

To resolve the issue of applying association rules to software project data described in Section 2, the proposed method handles quantitative variables using methods S1 and S2, as follows. **S1** is an extension of the association rules that uses statistics of a quantitative variable (mean and standard deviation) without conversion of the consequent part B. **S2** can be applied for one or more quantitative variables in antecedent part A. **S2** finds optimal fine-grained partitions by logically ORing the pre-determined partitions.

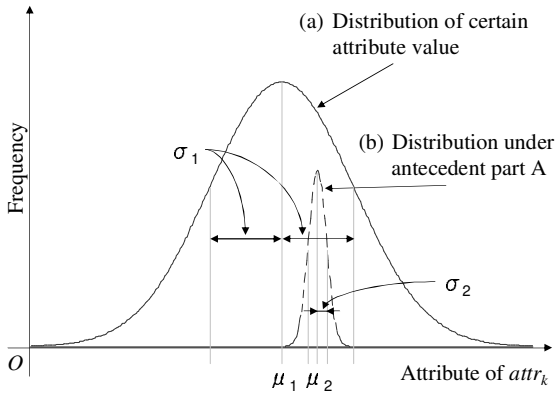


Figure 1 Distributions of attribute value

[S1] Extension of consequent part

S1 uses the attribute, the mean value, and the standard deviation of a quantitative variable in the consequent part B to create an extended association rule expressed as $A \Rightarrow attr_k(\mu, \sigma)$,

where $\mu = \frac{1}{l}\sum p_{ik} (1 \le i \le l)$, $\sigma = \sqrt{\frac{1}{l}\sum (p_{ik} - \mu)^2} (1 \le i \le l)$,

$l = |A \subset P|$.

The analyst specifies $attr_k$ for a rule mining. Rules are mined by calculating the mean and standard deviation of $attr_k$ in projects that meet antecedent A. An example would be "<industry, finance> ⇒ SLOC (84304, 163.565)."

We define the indicators below (lift of mean and lift of standard deviation) by comparing the means and standard deviations of all items (projects).

Lift of mean

The lift of mean is μ divided by the mean of the k^{th} attribute of all projects.

$$\text{lift of mean} = \frac{\mu}{\frac{\sum p_{ik}}{n}} (1 \le i \le n)$$

Lift of standard deviation

Similarly, lift of standard deviation $= \frac{\sigma}{\sqrt{\frac{\sum (p_{ik} - \mu)^2}{n}}} (1 \le i \le n)$

For example, given a quantitative rule "<development language, C> ⇒ productivity (2.0, 0.864)," if the mean productivity of all projects is 0.5, then the lift of mean is 2.0 / 0.5 = 4.0. The higher this value, the greater the effect of the antecedent is on the consequent in this rule.

Figure1 shows an example to explain lift of standard deviation. Solid line (a) is distribution of p_{ik} of all projects ($1 \le i \le n$). Dotted line (b) is distribution of p_{ik} of projects that meets antecedent part A ($A \subset P$). Lift of standard deviation is the ratio of σ_2 to σ_1. In this case, lift of standard deviation smaller than 1 ($\sigma_2 / \sigma_1 < 1$) indicates that situations expressed by the antecedent part A are drivers for smaller deviation. Enhancement of situations expressed by A may lead to smaller deviation of values of k^{th} attribute.

[S2] Partitioning and joining via conversion for antecedent part

S2 is applied to the antecedents part A. Using the method proposed by Srikant et al [13], quantitative variables are divided into multiple partitions that are converted into categories. It mines association rules from pre-converted categories, searches for rules in the obtained rule set that can join partitions, and ORs them to join the converted partitions. It is expected that the optimum partitioning will be found by creating a sufficiently large number of partitions. There are two partitioning methods, as described below. Both create $d (d \le n)$ partitions.

(1) For a given quantitative variable $attr_k$, divide v_{ik} into d equal parts. V_{lk} is a set partitioning the elements of V_k into d parts, where

$V_{lk} = \{< attr_k, v_{ik} > \in V_k | v_{ik} \ge v_{1k} + u(l-1)) \wedge v_{ik} < (v_{1k} + ul)\}$ $(1 \le l \le d)$ and

$u = \frac{v_{1k} - v_{n_k k}}{d}$.

(2) Partition the values so that as close as possible to an equal number of v_{ik} are in each interval.

V_{lk} is a set partitioning the elements of V_k into d parts, where

$V_{lk} = \{< attr_k, v_{(l-1)\cdot u_l + 1} >, ..., < attr_k, v_{lu_l} >\}$ $(1 \le l \le d)$

$$u_l = \begin{cases} \frac{n}{d} & (l = 1) \\ \frac{n - \sum u_i (1 \le i \le l-1)}{d - l} & (l \ne 1) \end{cases}$$

Quantitative variables are split into partitions V_k and converted. The discrete values of the mined rules meeting the following criteria are logically ORed and joined, and the support and confidence are recalculated.

Pairs in the mined rules meeting the following criteria are found:

$V_{lk} \wedge A' \Rightarrow B$, $V_{(l+1)k} \wedge A' \Rightarrow B (1 \le l \le d-1)$; and the logical OR (∨) is used to join V_{lk} and $V_{(l+1)k}$, like so:

$(V_{lk} \vee V_{(l+1)k}) \wedge A' \Rightarrow B (1 \le l \le d-1)$

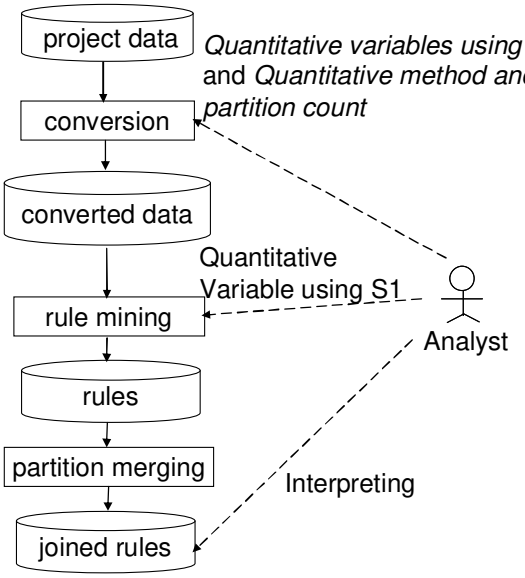
Although the antecedents of rules are joined, their consequents are not. This process continues until no joinable rules are found. If two rules are joined, the support, the lift of mean, and the lift of standard deviation are recalculated as shown below.

Support after joining

$support((V_{lk} \vee V_{(l+1)k}) \wedge A' \Rightarrow B) =$
$support(V_{lk} \wedge A' \Rightarrow B) + support(V_{(l+1)k} \wedge A' \Rightarrow B)$

Table 3 Examples of Mined Rules

	Rule	Support	Lift of mean	Lift of standard deviation
R1	(customer = existing customer) ∧ (target industrial = experienced) ⇒ ratio of outsourcing(mean: 0.368, standard deviation: 0.113)	0.216	1.510	0.832
R2	(development type = new development) ∧ (maximum number of staffs = smallest (1)) ⇒ ratio of outsourcing (mean: 0.118, standard deviation: 0.0630)	0.216	0.482	0.463
R3	(customer = existing customer) ∧ (use of commercial packages = without using) ∧ (proportion of staff month(coding and unit testing phase) = large (5 ∨ 6)) ⇒ proportion of staff month (integration and system testing)(mean: 0.210, standard deviation: 0.0352)	0.216	0.785	0.353
R4	(development type = new development) ∧ (target industrial = experienced) ∧ (outsourcer = second or later trading) ∧ (ratio of outsourcing = large (5 ∨ 6)) ⇒ proportion of staff month (integration and system testing)(mean: 0.262, standard deviation: 0.150)	0.216	0.979	1.51



S1 and **S2** are not mutually exclusive methods. If the target data has multiple quantitative variables, it is possible to specify one quantitative variable as a consequent to be applied by **S1**, and apply **S2** to the rest of the quantitative variables (appearing in the antecedent). In other words, it is possible to do the following: $A_1 \vee A_2 \Rightarrow attr_k(\mu,\sigma)$.

Here, $\mu = \dfrac{\sum p_{i_1 k} + \sum p_{i_2 k}}{l_1 + l_2}$

and $\sigma = \sqrt{\dfrac{\sum (p_{i_1 k} - \mu)^2 + \sum (p_{i_2 k} - \mu)^2}{l_1 + l_2}}$

where $(1 \le i_1 \le l_1, 1 \le i_2 \le l_2)$.

Note, however, that $l_1 = |A_1 \subset P|$, $l_2 = |A_2 \subset P|$.

3.3 Procedure

Figure 2 shows the procedure for extended association rule mining. The cylinders in the figure represent the data, and the squares represent processing. The solid arrows in the figure represent the flow of data, and the dotted arrows represent operations by the analyst. Processing proceeds in the following sequence: conversion, rule mining, and partition joining.

The analyst specifies the quantitative variables to use with **S2**, assigns a partition count *d* and partition method, and executes the "conversion" procedure. Conversion categorizes quantitative variables into discrete data (converts them to ordinal measurements). The analyst then executes the "mine rules" procedure specifying which quantitative variable to use with **S1** and a minimum support level. If the analyst has specified any quantitative variables for **S2**, the procedure "partition joining" merges rules with adjacent partitions. If the procedure finds rules capable of joining partitions, the rules are combined via a logical OR. When joining, the support, lift of mean, and lift of standard deviation of rules are re-calculated.

4. Related Research

Fukuda et al [6] have proposed a method for mining association rules including quantitative variables as antecedents. This method is capable of calculating for intervals; for example, given the quantitative variable age, it is able to calculate the values x_1, x_2 for which the rule "age interval [x_1, x_2] ⇒ purchased given service A" has the highest

support. Reference [5] also extends this method so that it can handle two quantitative variables. Although these methods can only mine rules with quantitative variables in the antecedent, they are one solution to the issue of handling quantitative variables in association-rule analysis. The present research can also calculate the interval with higher support as Fukuda et al do, by converting quantitative variables into qualitative variables (ordinal measurement), and joining rules via logical ORs.

A number of case studies have reported association-analysis methods for software project actual data. Amasaki et al [2] evaluate risk items for each development phase from collected questionnaires, and conduct association analysis for project-confusion factors (whether development budgets or deadline standards will be overrun), with the goal of revealing the factors leading to disorder in software-development projects. Their analysis data, however, does not include quantitative variables, and effective rules are only mined within the scope of conventional association analysis.

Song et al [12] mine association rules from defect data logged during development (type of defect cause, correction effort, etc.) to predict defects with a high likelihood of simultaneous occurrence and predict defect-correction effort (staff-hours). Although they convert correction effort, a quantitative variable, into ordinal form, the discrete partitions are hard-wired into four categories: 1 hour or less, 1 hour to one day, one to three days, and longer than three days. Applying S2 to Song et al's data should enable more fine-grained categories to be obtained. Additionally, method S1 could enable access to new knowledge by mining rules with mean correction effort and standard deviation in the consequent.

5. Conclusions

This paper proposes a method to mine rules from software engineering data repositories that contain a number of quantitative attributes such as staff months, LOC, defect density, test case density, and outsourcing cost. The proposed method extends conventional association analysis methods to treat quantitative variables in two ways. First, the proposed method extends association rules to include a single specified quantitative variable's mean value and standard deviation in the consequent part. Second, to treat other quantitative variables, the proposed method divides quantitative variables into contiguous fine-grained partitions appearing in the antecedent in preprocessing. Partitions next to each other are joined after rules are mined.

Since consequent parts of mined rules show distributions in the cause of antecedent parts, finding a difference of distribution leads to quick cause identifications, systematic process improvements, better planning, and more precise estimations. If a certain antecedent part increases the mean value of the consequent undesirably, eliminating the situation expressed in the antecedent part will decrease the mean value of the consequent part, providing quick cause identification and systematic process improvement. If a certain antecedent part increases the standard deviation of the consequent part, we can consider the variation expressed in the antecedent during planning and estimation in the project to provide better planning and estimations that are more precise.

The proposed method can be applied to very large software-project repositories including missing data. Furthermore, the proposed method can be applied to existing repositories. We are planning further investigation on larger software project repositories and other kinds of repository.

Acknowledgements

This work is supported by the Comprehensive Development of e-Society Foundation Software program of the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

[1] Agrawal R., Imielinski T., Swami A.,: Mining Association Rules between Sets of Items in Large Databases, Proceedings of ACM SIGMOD Conference on Management of Data, pp. 207-216 (1993).

[2] Amasaki S. , Hamano Y. , Mizuno O. , and Kikuno T. , "Characterization of Runaway Software Projects Using Association Rule Mining," In Proceedings of 7th International Conference on Product Focused Software Process Improvement, pp.402-407, June 2006.

[3] Boehm B. W., Software Engineering Economics, Prentice Hall, 1981.

[4] Chillarege R., Bhandari I.S., Chaar J.K., Halliday M.J., Moebus D.S., Ray B.K., Wong M.Y.: Orthogonal Defect Classification-A Concept for In-Process Measurements, IEEE Transaction on Software Engineering, Vol. 18, No. 11, pp. 943-956 (1992).

[5] Fukuda T., Morimoto Y., Morishita S., Tokuyama T., : Data Mining Using Two Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization, In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 13-23 (1996).

[6] Fukuda T., Morimoto Y., Morishita S., Tokuyama T., : Mining Optimized Association Rules for Numeric Attributes In Proceedings of the 5th ACM SIGACT-SIGMOD SIGART Symposium on Principles of Database Systems, pp. 182-191, (1996).

[7] IPA SEC, http://www.ipa.go.jp/english/sec/first.html

[8] International Software Benchmarking Standards Group Repository Information,

http://www.isbsg.org/isbsg.nsf/weben/Repository%20info

[9] Ramamoorthy C. V.; Bastani F. B.,: Software reliability - Status and perspectives, IEEE Transactions on Software Engineering. Vol. 8, No. 4, pp. 354-371. July 1982

[10] She R., Chen F., Wang K., Ester M., Gardy J.L., Brinkman F.L.: Frequent-Subsequence-Based Prediction of Outer Membrane Proteins, Proceedings of 9th ACM SIGKDD International conference on Knowledge Discovery and Data Mining, pp. 436-445, (2003).

[11] Shepperd M., Schofield C., : Estimating Software Project Effort Using Analogies, IEEE Transaction on Software Engineering Vol. 23, No. 12, pp. 736-743 (1997).

[12] Song Q. , Shepperd M., Michelle Cartwright, and Carolyn Mair: Software Defect Association Mining and Defect Correction Effort Prediction, IEEE Transaction on Software Engineering, Vol. 32, No. 2, pp. 69-82, (2006).

[13] Srikant R., Agrawal R., : Mining Quantitative Association Rules in Large Relational Tables, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 1-12, (1996)

[14] Srinvasan K., Fischer D., : Machine Learning Approaches to Estimating Software Development Effort, IEEE Transaction on Software Engineering, Vol.21, No.2, pp. 126-137 (1995).

[15] Yang Q., Zhang H.H., Li T., "Mining Web Logs for Prediction Models in WWW Caching and Prefetching, Proceedings of Seventh ACM SIGKDD International Conference of Knowledge Discovery and Data Mining, pp. 473-478, (2001).